第一章　等切面曲线和相似曲线

W. Gander, S. Bartoň and J. Hřebíček

1.1　引言

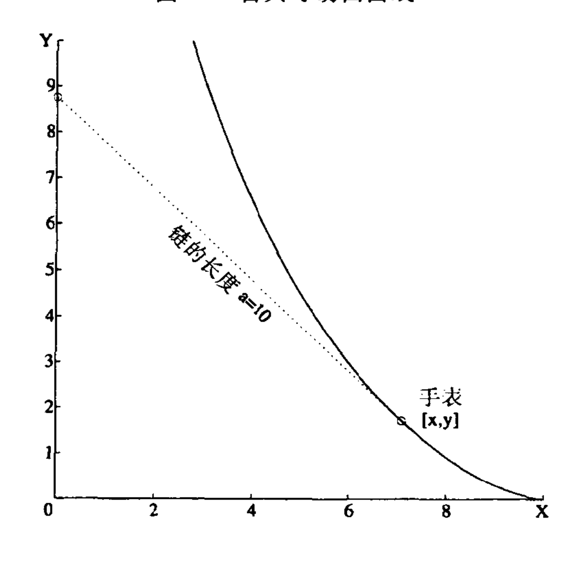
在这一章里，我们将用 MATLAB 解两个相似的微分方程系统．首先，我们推广古典的等切面曲线问题，计算小孩拉玩具的轨迹．然后计算狗攻击慢跑者的轨迹．我们还用 MATLAB 显示这些运动轨迹．

1.2　古典等切面曲线

在 17 世纪， Gottfried Wilhelm Leibniz 曾讨论下面问题，参见 [2,1]: 将一个手表系在一条链子上，当沿直线拉此链的一端时，在平面上，此手表所描述的轨迹是什么？

假设 a 为此链的长度，手表看成一个点，如果假设手表开始位于 x 轴的点 $(a,0)$ 处，我们从原点开始在 y 轴的正方向上拉，则容易求解此问题 [2],(参见图 1.1).

图 1.1　古典等切面曲线



从图 1.1, 我们可得函数 $y(x)$ 的微分方程：

$$y' = -\frac{\sqrt{a^2 - x^2}}{x}. \tag{1.1}$$

为了解方程 (1.1), 我们只需积分：

```
>  assume(a>=0);
>  y:= -int(sqrt(a^2-x^2)/x,x)+c;
```

$$y := -\sqrt{a^{\sim 2} - x^2} + a^{\sim}\operatorname{arctanh}\left(\frac{a^{\sim}}{\sqrt{a^{\sim 2} - x^2}}\right) + c \tag{1.2}$$

当执行不定积分时， MAPLE 不包含常数. 因此我们增加一个积分常数 c. 利用 $\lim_{x\to a} y(x) = 0$. 我们能确定它的值:

```
> c:= solve(limit(y,x=a),c);
```

$$c := \frac{1}{2} I\, a^{\sim}\, \pi$$

利用 MAPLE 5.3, 我们从 int 直接得到一个实表达式. 因为对于实表达式，此常数为 0, 我们不必计算一个复的积分常数.

```
> yold:= -sqrt(a^2-x^2)+a*arctanh(sqrt(a^2-x^2)/a);
```

$$yold := -\sqrt{a^{\sim 2} - x^2} + a^{\sim}\operatorname{arctanh}\left(\frac{\sqrt{a^{\sim 2} - x^2}}{a^{\sim}}\right) \tag{1.3}$$

为了证明两个结论相同，我们把它们相减并转成指数对数形式，化简并展开

```
> e:= expand(simplify(convert((y-yold)/a,expln)));
```

$$e := -\frac{1}{2}\ln(\sqrt{a^{\sim 2} - x^2} - a^{\sim}) + \frac{1}{2} I\,\pi + \frac{1}{2}\ln(-\sqrt{a^{\sim 2} - x^2} + a^{\sim})$$

在 MAPLE 中，不可能再化简此式. 但是应该注意，第一个对数函数中的量是第二个的负值. 因此，如果用一个正的未知数 d 代替 $a - \sqrt{a^2 - x^2}$, 类似地用 $-d$ 代替 $\sqrt{a^2 - x^2} - a$, 则 MAPLE 能化简这个表达式:

```
> assume(d>0);
> simplify(subs({(a^2-x^2)^(1/2)-a=-d,-(a^2-x^2)^(1/2)+a=d},e));
```

$$0$$

假设被拉物体的初始位置在 y 轴上的点 $(0, a)$, 我们从原点开始拉，但这次在 x 轴的正方向上.

考虑在物体的轨迹上的点 $(x, y(x))$. 在 x 轴上，此链的端点是 $(x - y(x)/y'(x), 0)$, 即切线与 x 轴的交点 (与 Newton 迭代一步所得的点相同 !). 所以，具有常数长度 a 的此链，导出微分方程

$$\frac{y(x)^2}{y'(x)^2} + y(x)^2 = a^2\ , \tag{1.4}$$

不能通过求积分直接解它. 因此，需要调用微分方程的求解程序 dsolve,

```
> restart;
> assume(a>0);
> eq := (y(x)/diff(y(x), x))^2 + y(x)^2 = a^2;
```

$$eq := \frac{\mathrm{y}(x)^2}{(\frac{\partial}{\partial x}\mathrm{y}(x))^2} + \mathrm{y}(x)^2 = a^{\sim 2}$$

```
> p:=dsolve(eq,y(x));
```

$$p := \sqrt{-\mathrm{y}(x)^2 + a\tilde{}^2} - a\tilde{}\,\mathrm{arctanh}(\frac{a\tilde{}}{\sqrt{-\mathrm{y}(x)^2 + a\tilde{}^2}}) + x = _C1,$$
$$-\sqrt{-\mathrm{y}(x)^2 + a\tilde{}^2} + a\tilde{}\,\mathrm{arctanh}(\frac{a\tilde{}}{\sqrt{-\mathrm{y}(x)^2 + a\tilde{}^2}}) + x = _C1$$

并得到两个解. 从初始条件 $y(0) = a$ 和物理学, 对于 $a > 0$, 可得 $y(x) > 0$ 和 $y'(x) < 0$. 因此, 第一个解是我们问题的正确答案:

```
> p[1];
```

$$\sqrt{-\mathrm{y}(x)^2 + a\tilde{}^2} - a\tilde{}\,\mathrm{arctanh}(\frac{a\tilde{}}{\sqrt{-\mathrm{y}(x)^2 + a\tilde{}^2}}) + x = _C1$$

我们得到隐式的解 $y(x)$. 为了求积分常数 _ C1, 我们利用约束 $\lim_{x\to 0} y(x) = a$

```
> _C1:= limit(lhs(subs({x=0,y(x)=y},p[1])),y=a);
```

$$_C1 := \frac{1}{2} I\, a\tilde{}\, \pi$$

于是, 解满足方程

$$\sqrt{-y(x)^2 + a\tilde{}^2} - a\tilde{}\,\mathrm{arctanh}(\frac{a\tilde{}}{\sqrt{-y(x)^2 + a\tilde{}^2}}) + x = \frac{1}{2} I\ a\tilde{}\,\pi$$

由 MAPLE 5.3, 可得到实表达式

$$\sqrt{a^2 - y(x)^2} - a\,\mathrm{arctanh}\left(\frac{\sqrt{a^2 - y(x)^2}}{a}\right) + x = 0$$

当然, 我们也可在方程 (1.2) 和 (1.3) 中, 互换变量 x 和 y 得到这些方程. 值得注意的是, 因为对 $x = 0$, 有奇异性 $y'(0) = \infty$, 所以用数值方法解方程 (1.4) 是困难的.

1.3 小孩和玩具

让我们解决一个更一般的问题, 假设一个小孩在平面上沿一曲线行走, 此曲线由两个时间的函数 $X(t)$ 和 $Y(t)$ 确定.

假设此小孩借助长度为 a 的硬棒, 拉或推某玩具. 当此小孩沿曲线行走时, 我们计算玩具的轨迹. 设 $(x(t), y(t))$ 是玩具的位置.

从图 1.2 可得下列方程:

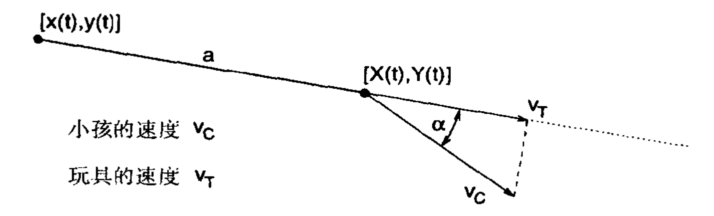
1. $(X(t), Y(t))$ 与 $(x(t), y(t))$ 之间的距离总是硬棒的长度, 于是

$$(X - x)^2 + (Y - y)^2 = a^2. \tag{1.5}$$

2. 玩具总是在硬棒的方向上运动, 因此, 两个位置的差向量是玩具的速度向量的倍数, $\mathbf{v}_T = (\dot{x}, \dot{y})^{\mathrm{T}}$:

$$\begin{pmatrix} X - x \\ Y - y \end{pmatrix} = \lambda \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \quad \text{其中} \quad \lambda > 0. \tag{1.6}$$

图 1.2 速度 $\mathbf{v_C}$ 和 $\mathbf{v_T}$



3. 玩具的速度依赖于小孩的速度向量 $\mathbf{v_C}$ 的方向. 例如, 假设小孩在半径为 a（硬棒的长）的圆上行走. 在此特殊情况下, 玩具停留在此圆的圆心, 根本不运动（这是第一个例子的最后状态, 参见图 1.3).

从图 1.2 可知, *小孩的速度 $\mathbf{v_C}$ 在硬棒上的投影的模是玩具的速度 $\mathbf{v_T}$ 的模.*
将方程 (1.6) 代入方程 (1.5) 中, 可得

$$a^2 = \lambda^2(\dot{x}^2 + \dot{y}^2) \;\rightarrow\; \lambda = \frac{a}{\sqrt{\dot{x}^2 + \dot{y}^2}}.$$

于是,

$$\frac{a}{\sqrt{\dot{x}^2 + \dot{y}^2}} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} X - x \\ Y - y \end{pmatrix}. \tag{1.7}$$

为了得到 $\dot{x}$ 和 $\dot{y}$, 我们要解方程 (1.7). 因为玩具的速度的模 $|\mathbf{v}_T| = |\mathbf{v}_C|\cos\alpha$, 见图 1.2, 这由下面步骤可得到:

- 标准化差向量 $(X - x, Y - y)^{\mathrm{T}}$, 可得单位长的向量 $\mathbf{w}$.
- 确定 $\mathbf{v}_C = (\dot{X}, \dot{Y})^{\mathrm{T}}$ 在 $\mathbf{w}$ 生成的子空间上的投影. 因为 $\mathbf{v}_C^{\mathrm{T}}\mathbf{w} = |\mathbf{v}_C||\mathbf{w}|\cos\alpha$ 和 $|\mathbf{w}| = 1$, 所以这就是内积 $\mathbf{v}_C^{\mathrm{T}}\mathbf{w}$.
- $\mathbf{v}_T = (\dot{x}, \dot{y})^{\mathrm{T}} = (\mathbf{v}_C^{\mathrm{T}}\mathbf{w})\mathbf{w}$.

现在, 我们能用 MATLAB 写出函数, 求解微分方程系统.

算法 1.1 函数 f

```
function zs = f(t,z)
%
[X Xs Y Ys] = child(t);
v =[Xs; Ys];
w =[X-z(1); Y-z(2)];
w = w/norm(w);
zs = (v'*w)*w;
```

函数 f 调用一个函数 child, 对于给定的时间 t, 它返回小孩的位置 $(X(t), Y(t))$ 和速度 $(Xs(t), Ys(t))$. 例如, 考虑小孩在圆 $X(t) = 5\cos t; Y(t) = 5\sin t$ 上行走, 此时相应的函数 child 是:

算法 1.2 函数 Child

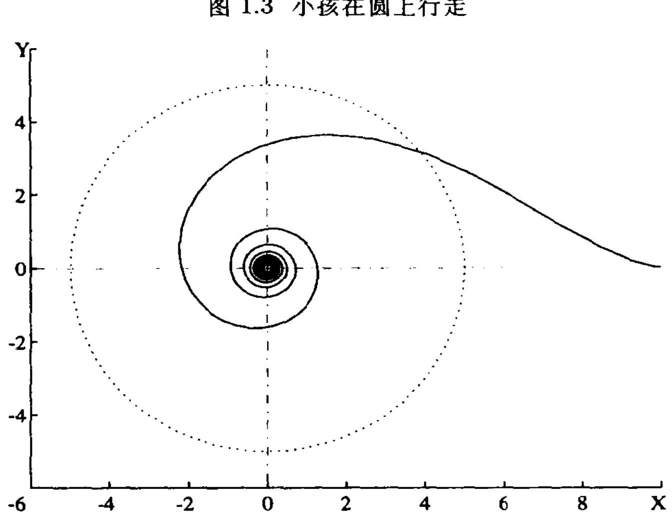
```
function [X, Xs, Y, Ys] = child(t);
%
```

```
X  =  5*cos(t);  Y  =  5*sin(t);
Xs = -5*sin(t);  Ys =  5*cos(t);
```

MATLAB 提供两个 M 文件 ode23 和 ode45, 用于求积微分方程. 在下面的主程序中, 我们将调用这些函数之一, 并定义初始条件 (注意, 当 $t=0$ 时, 小孩在点 (5,0) 处和玩具在点 (10,0) 处):

```
>> % main1.m
>> y0 = [10 0]';
>> [t y]  = ode45('f',[0 100],y0);
>> clf; hold on;
>> axis([-6 10 -6 10]);
>> axis('square');
>> plot(y(:,1),y(:,2));
```

图 1.3 小孩在圆上行走



如果绘制两列 y, 则可得玩具的轨迹 (参见图 1.3), 而且在同一图中, 用语句加进小孩的曲线:

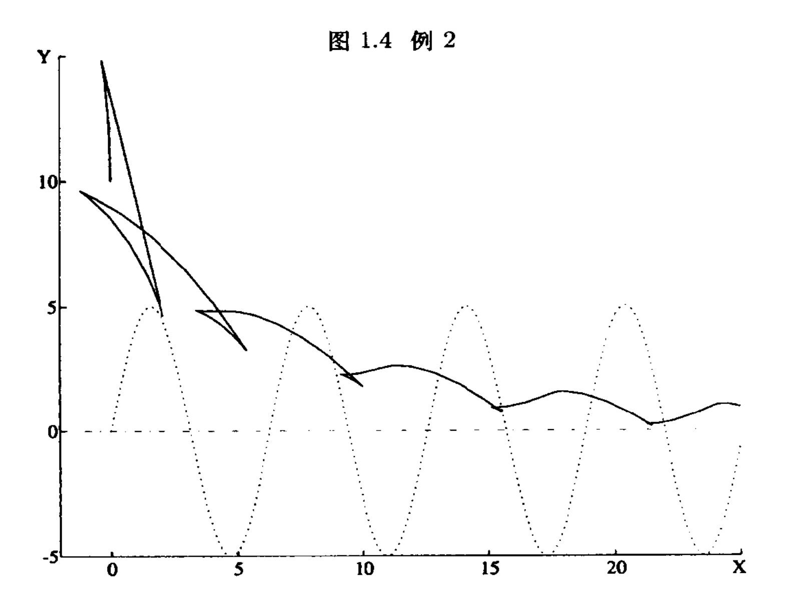
```
>> t = 0:0.05:6.3;
>> [X, Xs, Y, Ys] = child(t);
>> plot(X,Y,':');
>> hold off;
```

注意, 在程序中, 硬棒的长 a 没有明显出现; 它由玩具的位置(初始条件),和 $t=0$ 时小孩的位置(函数child) 隐含地定义.

我们用几个例子来结束本节. 假设小孩沿 sin 函数的图形行走: $X(t)=t$ 和 $Y(t)=5\sin t$, 小孩的曲线用虚线表示, 初始条件 $x(0)=0$ 和 $y(0)=10$, 我们得到图 1.4.

另一个例子, 小孩在圆 $X(t)=5\cos t, Y(t)=5\sin t$ 上行走, 初始条件 $x(0)=0$ 和 $y(0)=10$, 我们得到一个象花的玩具轨迹 (参见图 1.5).

图 1.4 例 2



1.4 慢跑者和狗

我们考虑下面问题：为了每天锻炼，一个慢跑者在平面上沿着他喜欢的路径跑步，突然一只狗攻击他，这只狗以恒定速率 w 跑向慢跑者，计算狗的轨迹.

狗的轨迹具有如下性质：在任意时刻，狗的速度向量都指向它的目标慢跑者. 假设慢跑者在某路径上跑步，他的运动由两个函数 $X(t)$ 和 $Y(t)$ 描述.

假设当 $t=0$ 时，狗是在点 (x_0, y_0) 处，在时刻 t 时，它的位置是 $(x(t), y(t))$, 下列方程成立：

1. $\dot{x}^2+\dot{y}^2=w^2$: 狗以恒定速率跑.
2. 狗的速度向量平行于慢跑者与狗的位置的差向量：

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \lambda \begin{pmatrix} X-x \\ Y-y \end{pmatrix} \quad \text{其中 } \lambda > 0.$$

如果将它代入第一个方程，我们得到

$$w^2 = \dot{x}^2+\dot{y}^2 = \lambda^2 \left\| \begin{pmatrix} X-x \\ Y-y \end{pmatrix} \right\|^2.$$

求解此方程，可得 λ：

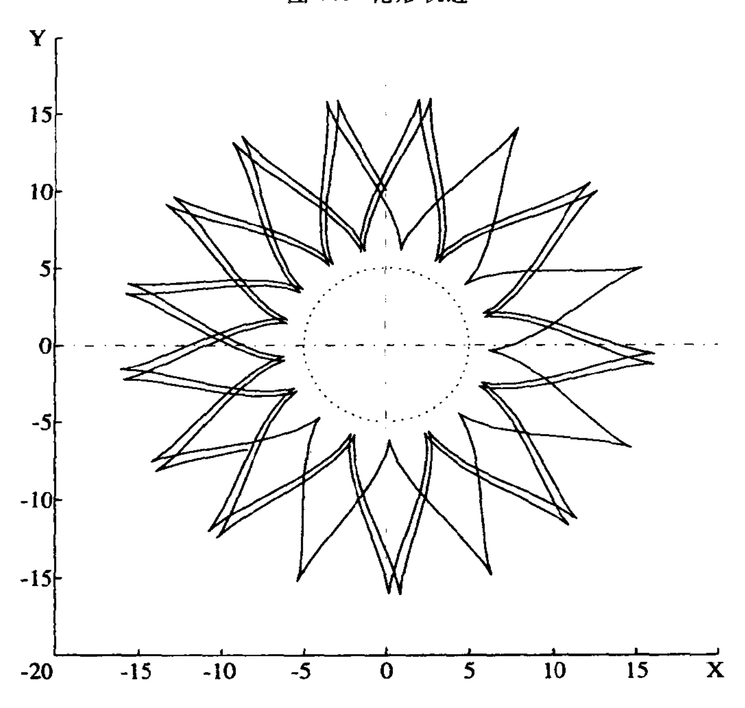
$$\lambda = \frac{w}{\left\| \begin{pmatrix} X-x \\ Y-y \end{pmatrix} \right\|} > 0.$$

最后，将此式代入第二个方程，可得狗的轨迹的微分方程：

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \frac{w}{\left\| \begin{pmatrix} X-x \\ Y-y \end{pmatrix} \right\|} \begin{pmatrix} X-x \\ Y-y \end{pmatrix}. \tag{1.8}$$



图 1.5 花形轨迹



我们将利用 M 文件 ode23.m 或 ode45.m, 求积微分方程系统. 注意, 当狗达到慢跑者时, 系统 (1.8) 有一个奇异点. 此时差向量的范数变为 0, 我们必须停止积分. 上面提到的求积微分方程的 MATLAB 函数, 要求输入独立变量的一个区间. 现在, MATLAB 5.0 也可定义另一个求积终止准则, 它不同于给定独立变量的一个上界, 而是通过检查函数的零交点, 来终止积分. 在我们的例子中, 当狗达到慢跑者, 即 $\|(X-x, Y-y)\|$ 变成很小时, 将终止积分. 为此, 在 M 函数 dog.m 中, 我们必须增加第三个输入参数和两个新的输出参数. 积分器 ode23 或 ode45, 以两种方法调用此函数: 第一种方法是取消第三个参数, 函数只返回参数 zs : 狗的速率. 第二种方法是, 将关键字 'events' 赋值给参数 flag, 此关键字告诉函数, 在第一个输出 zs 中返回零交点函数. 第二个输出 isterminal 是一个逻辑向量, 它告诉积分器, 当第一个输出的分量变成 0 时, 它们迫使过程终止. 具有这种性质的每一个分量, 在 isterminal 中, 用非零标记. 第三个输出参数 direction 也是一个向量, 它表示: 对 zs 的每个分量, 零交点是否仅被视为增加值 (direction = 1), 下降值 (direction = -1) 或两者 (direction = 0). 零交点的条件将在积分器里检查. dog 和主程序中, 必须声明狗的速率 w 是全局变量. M 函数 jogger.m 给出慢跑者的轨迹.

算法 1.3 函数 Dog

```
function [zs,isterminal,direction] = dog(t,z,flag);
%
global w  % w = speed of the dog
X= jogger(t);
h= X-z;
nh= norm(h);
```

```
    if nargin < 3 | isempty(flag) % normal output
       zs= (w/nh)*h;
    else
       switch(flag)
       case 'events'  % at norm(h)=0 there is a singularity
          zs= nh-1e-3;   % zero crossing at pos_dog=pos_jogger
          isterminal= 1; % this is a stopping event
          direction= 0;  % don't care if decrease or increase
       otherwise
          error(['Unknown flag: ' flag]);
       end
    end
```

主程序 main2.m 定义初始条件，调用 ode23 进行积分. 为了求积，我们必须提供时间 t 的上界.

```
>> %main2.m
>> global w
>> y0=[60;70];%initial condition,starting point of the dog
>> w = 10;     %w speed of the dog
>> options= odeset('RelTol',1e-5,'Events','on');
>> [t,Y] = ode23('dog',[0,20],y0,options);
>> clf; hold on;
>> axis([-10,100,-10,70]);
>> plot(Y(:,1),Y(:,2));
>> J =[];

>> for h=1:length(t),
>>      w = jogger(t(h));
>>      J = [J; w'];
>> end;
>> plot(J(:,1),J(:,2),':');
```

如果达到时间 t 的上界或狗赶上慢跑者，积分将终止. 对于后者，我们设置 ODE 选项的标志 Events 为 'on'，这告诉积分器检查，具有 flag ='events' 的被调用函数 dog 的零交点. 调用 ode23 后，变量 Y 包含有两个函数 $x(t)$ 和 $y(t)$ 的值的表. 用语句 plot(Y(:,1), Y(:,2))，我们画出狗的轨迹. 为了显示慢跑者的轨迹，我们需要用向量 t 和函数 jogger 计算它.

现在，我们计算几个例子. 首先，假设慢跑者沿 x 轴跑步:

算法 1.4 第一个慢跑者例子

```
function s = jogger(t);
s  = [8*t; 0];
```

在上面的主程序里，我们选取狗的速率为 $w = 10$，因而 $X(t) = 8t$，慢跑者是较慢的. 如图 1.6 所示，狗抓住慢跑者.

如果想指出慢跑者有麻烦的位置，(也许设立一个小纪念物，) 我们利用下面文件 cross.m

算法 1.5 画十字标

```
function cross(Cx,Cy,v)
% draws at position  Cx,Cy  a cross of height 2.5v
% and width 2*v
Kx = [Cx Cx Cx Cx-v Cx+v];
Ky = [Cy Cy+2.5*v Cy+1.5*v Cy+1.5*v Cy+1.5*v];
plot(Kx,Ky);
plot(Cx,Cy,'o');
```

图中的十字标是由主程序增加语句

```
>> p = max(size(Y));
>> cross(Y(p,1),Y(p,2),2)
>> hold off;
```

所产生. 下一个例子显示慢跑者转向的位置并跑回家:

算法 1.6 第二个慢跑者例子

```
function s = jogger1(t);
%
if t<6, s = [8*t; 0];
else    s = [8*(12-t) ;0];
end
```

图 1.6 慢跑者沿直线 $y = 0$ 跑

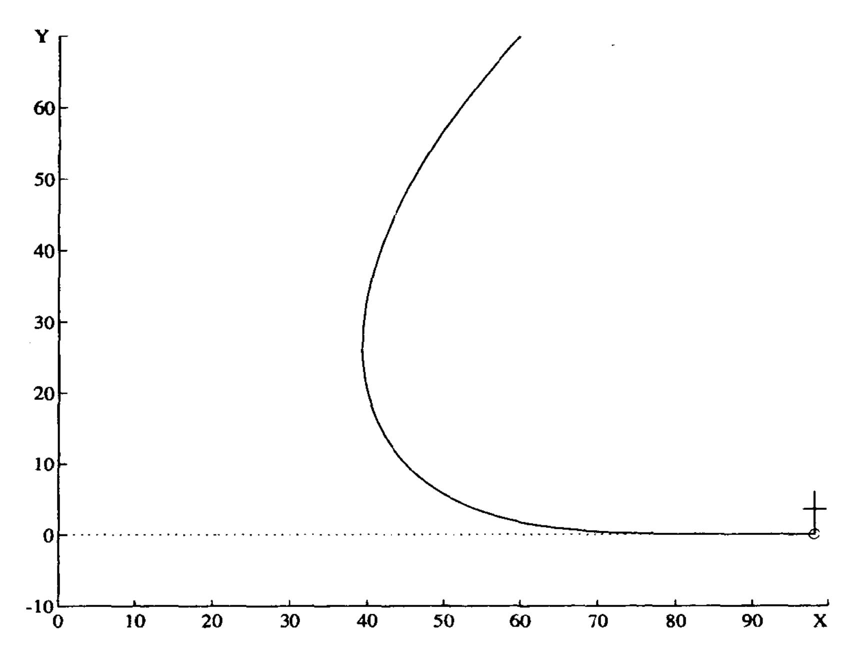
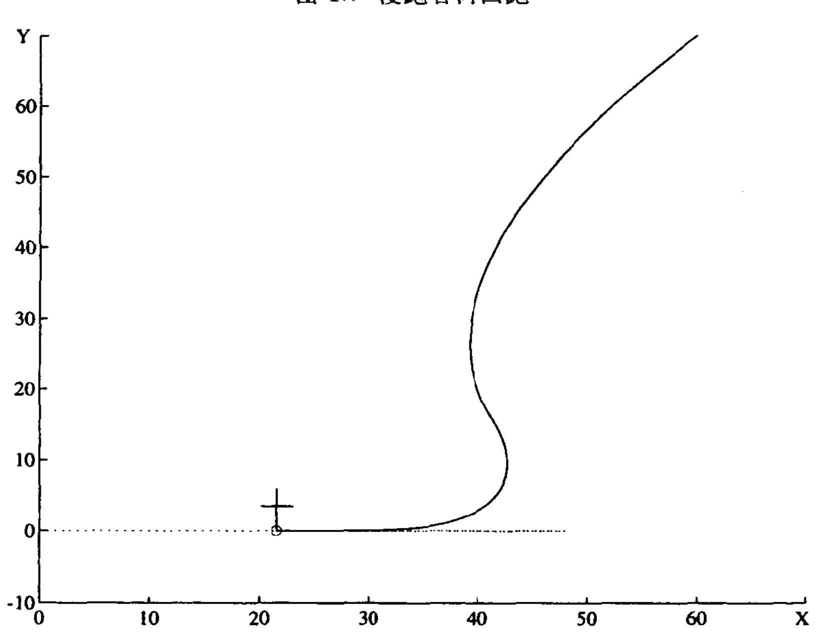


图 1.7 慢跑者向回跑



用前面相同的主程序，当 $t = 9.3$ 时，狗赶上慢跑者 (参见图 1.7). 现在，考虑一个更快的慢跑者在一个椭圆上跑步

算法 1.7 第三个慢跑者例子

```
function s = jogger2(t);
s    = [  10+20*cos(t)
          20 + 15*sin(t)];
```

如果狗跑得快 ($w = 19$), 当 $t = 8.97$ 时，它赶上慢跑者 (参见图 1.8). 最后，考虑一只老的、慢的狗 ($w = 10$), 它在一个椭圆上跑，并且努力赶上慢跑者. 然而，狗不在椭圆上的某点等待，而是在它的目标后面跑 (太慢), 我们能看到一个稳定的状态，狗在椭圆内一个闭轨迹上跑 (参见图 1.9).

1.5 用 MATLAB 显示运动

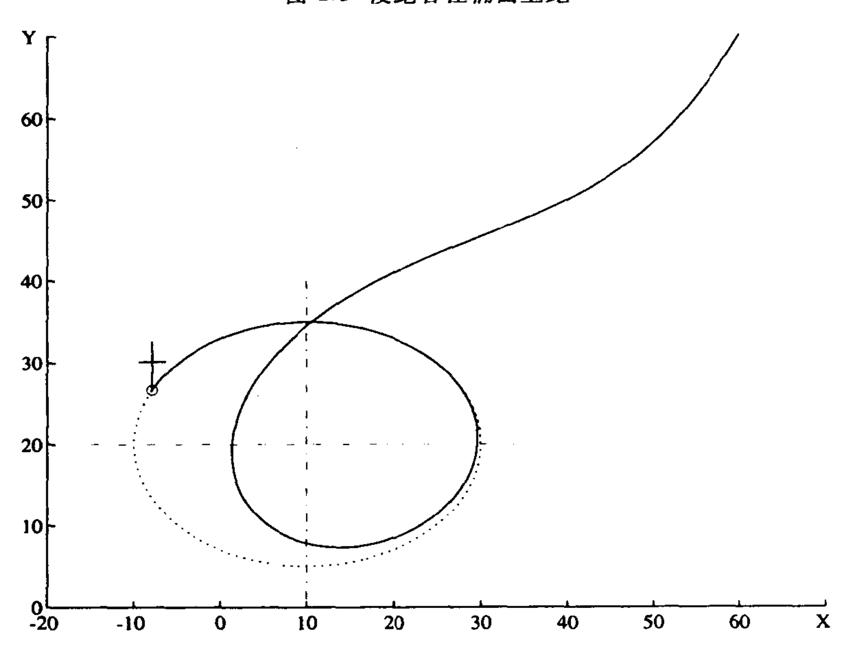
同时显示小孩和玩具，或狗和慢跑者的运动，而不只是画出它们静态的轨迹，这样效果会更好，这可用 MATLAB 的图形句柄命令. 关于小孩和玩具的主程序如下：

```
>> %main3.m
>> y0 = [0 20]';
>> options= odeset('RelTol',1e-10);
>> [t y] = ode45('f', [0 40], y0, options);
>> [X, Xs, Y, Ys] = child (t);

>> xmin = min (min (X), min (y (:, 1)));
```



图 1.8 慢跑者在椭圆上跑

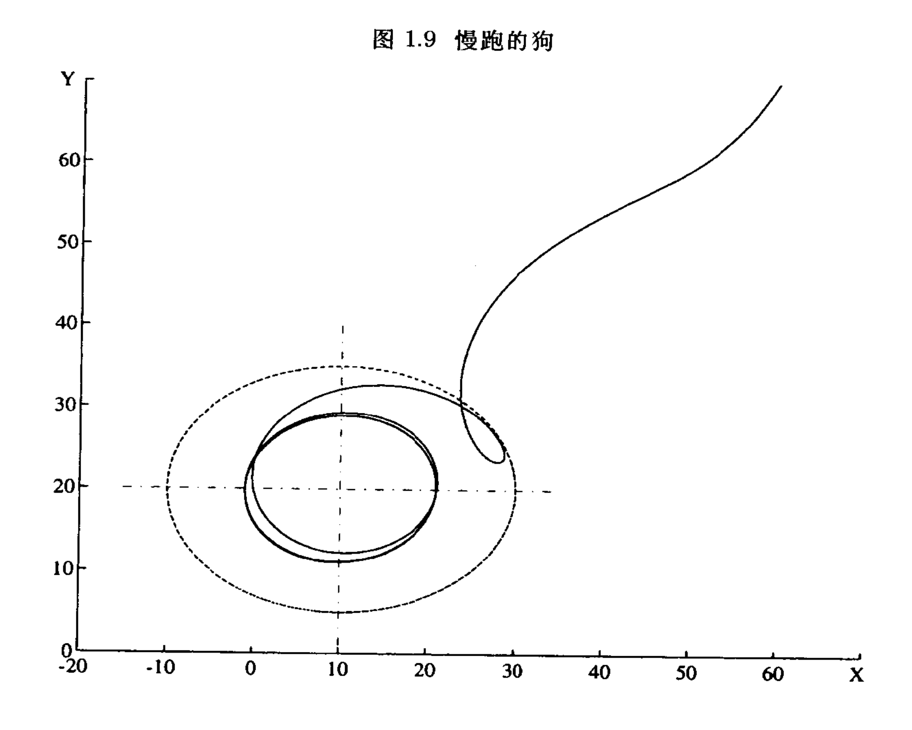



```
>> xmax = max (max (X), max (y (:, 1)));
>> ymin = min (min (Y), min (y (:, 2)));
>> ymax = max (max (Y), max (y (:, 2)));

>> clf; hold on;
>> axis ([xmin xmax ymin ymax]);
>> % axis('equal');
>> title ('The Child and the Toy.');
>> stickhandle=line('Color','yellow','EraseMode','xor',...
>>             'LineStyle','-','XData',[],'YData',[]);

>> for k = 1:length(t)-1,
>>    plot([X(k),X(k+1)],[Y(k),Y(k+1)],'-', ...
>>        'Color','yellow','EraseMode','none');
>>    plot([y(k,1),y(k+1,1)],[y(k,2),y(k+1,2)],'-',...
>>        'Color', 'green', 'EraseMode', 'none');
>>    set (stickhandle,'XData',[X(k+1),y(k+1,1)], ...
>>        'YData', [Y(k+1), y(k+1,2)]);
>>    drawnow;
>> end;
>> hold off;
```

图 1.9 慢跑的狗



对于联接棒的直线类型的图形对象，我们定义变量 stickhandle 为它的一个句柄．在循环内，我们画出小孩和玩具的轨迹新的部分，并移动棒的位置．用 drawnow 命令可即时画出这些图形，因此，我们能看到同时画出两个轨迹和棒．

在慢跑者和狗的情况下，我们不必定义句柄．我们所作的只是以适当的顺序，画出两个轨迹的各部分：

```
>> %main4.m
>> global w;
>> y0=[60;70];%initial conditions,starting point of the dog
>> w = 10;     %w speed of the dog
>> options= odeset('RelTol',1e-5,'Events','on');
>> [t,Y] = ode23('dog',[0,20],y0,options);

>> J = [];
>> for h= 1:length(t);
>>     w = jogger(t(h));
>>     J = [J; w'];
>> end

>> xmin = min (min (Y (:, 1)), min (J (:, 1)));
>> xmax = max (max (Y (:, 1)), max (J (:, 1)));
>> ymin = min (min (Y (:, 2)), min (J (:, 2)));
```

```
>> ymax = max (max (Y (:, 2)), max (J (:, 2)));
>> clf; hold on;
>> axis ([xmin xmax ymin ymax]);
>> % axis('equal');
>> title ('The Jogger and the Dog.');

>> for h = 1:length(t)-1,
>>   plot ([Y(h,1),Y(h+1,1)],[Y(h,2),Y(h+1,2)],'-', ...
>>         'Color', 'yellow', 'EraseMode', 'none');
>>   plot ([J(h,1),J(h+1,1)],[J(h,2),J(h+1,2)],':', ...
>>         'Color', 'green', 'EraseMode', 'none');
>>   drawnow;
>>   pause(1);
>> end;
>> hold off;
```

1.6 具有恒定速率的慢跑者

在本节中，我们仍考虑慢跑者的椭圆轨迹. 如果椭圆仍如算法 1.7 中所描述，并考虑 t 为时间变量，则慢跑者的速度不是恒定的. 令 $s(t)$ 是描述中心在 (m_1,m_2) 和主半轴为 a 和 b 的椭圆的参数：

$$X(s) = m_1 + a\cos(s), \quad Y(s) = m_2 + b\sin(s).$$

我们希望确定， s 是时间 t 的单调递增函数，使得对于等间隔的时间 t_i，在椭圆的边界上，点 $X(s(t_i)),Y(s(t_i))$ 之间也是等距离. 一个等价的条件是慢跑者的速度 $V(t)$ 为常数：

$$V = \sqrt{\left(\frac{\mathrm{d}X(s(t))}{\mathrm{d}t}\right)^2 + \left(\frac{\mathrm{d}Y(s(t))}{\mathrm{d}t}\right)^2} = const. \tag{1.9}$$

现在，为了得到 $s(t)$，先通过解方程 (1.9)，计算 $\mathrm{Ds} = \mathrm{d}s/\mathrm{d}t$:

```
> Xe := m1 + a*cos(s(t)):  Ye := m2 + b*sin(s(t)):
> Ve2 := diff(Xe, t)^2 +diff(Ye, t)^2 =V^2;
```

$$Ve2 := a^2\sin(\mathrm{s}(t))^2\,(\frac{\partial}{\partial t}\,\mathrm{s}(t))^2 + b^2\cos(\mathrm{s}(t))^2\,(\frac{\partial}{\partial t}\,\mathrm{s}(t))^2 = V^2$$

```
> Ds := solve(Ve2, diff(s(t), t));
```

$$Ds := \frac{V}{\sqrt{a^2\sin(\mathrm{s}(t))^2 + b^2 - b^2\sin(\mathrm{s}(t))^2}},\ -\frac{V}{\sqrt{a^2\sin(\mathrm{s}(t))^2 + b^2 - b^2\sin(\mathrm{s}(t))^2}}$$

因为慢跑者逆时针运动，我们必须选取第一个方程．微分方程没有解析解，所以我们将用 MATLAB. 数值地求解它．加上两个微分方程 (1.8), 我们得到一个三对微分方程的系统：

$$
\begin{aligned}
X(t) &= m_1 + a\cos(s(t)) \\
Y(t) &= m_2 + b\sin(s(t)) \\
\dot{x}(t) &= c(X(t) - x(t)) \\
\dot{y}(t) &= c(Y(t) - y(t)) \\
\dot{s}(t) &= \frac{V}{\sqrt{a^2\sin(s(t))^2 + b^2\cos(s(t))^2}} \\
c &= \frac{w}{\left\| \begin{pmatrix} X - x \\ Y - y \end{pmatrix} \right\|}.
\end{aligned}
$$

用函数 fkt(算法 1.8) 实现此系统，由算法 1.9 给出对应的主程序.

算法 1.8 具有常速度慢跑者的函数 fkt

```
function [ydot,isterminal,direction] = fkt(t,y,flag)
% system of differential equations
% for the jogger-dog problem
% where the jogger runs with constant
% velocity on an ellipse
global a b m c w
A =  cos(y(3)); B =  sin(y(3));
X = m(1) + a*A; Y = m(2) + b*B;
h = [X;Y] -y(1:2); nh = norm(h);
zs =  (w/nh)*h;

if nargin < 3 | isempty(flag) % normal output
    ydot = [zs;c/sqrt((a*B)^2+(b*A)^2)];
else
    switch(flag)
    case 'events' % at norm(h)=0 there is a singularity
        ydot= nh-(1e-3); % zero crossing at pos_dog=pos_jogger
        isterminal= 1;   % this is a stopping event
        direction= 0;    % don't care if decrease or increase
    otherwise
        error(['Unknown flag ''' flag '''.']);
    end
end
```

为了与以前的计算比较，我们选取方程 (1.9) 中的常数为第 1.4 节例子中慢跑者的平均速度：

$$V \equiv \overline{V(t)} = \frac{2}{\pi}\int_0^{\frac{\pi}{2}} V(t)\mathrm{d}t \equiv \frac{L}{T}.$$

```
> T := 2*Pi:
> L := evalf(4*int(sqrt(20^2*sin(f)^2 + 15^2*cos(f)^2),f = 0..Pi/2));
> V := evalf(L/T);
```

$$L := 110.5174608$$
$$V := 17.58940018$$

注意，如果执行算法 1.9, 则当 $t = 8.22834$ 时，狗赶上慢跑者，这比 1.4 节中的更早一点.

算法 1.9 具有常速慢跑者的主程序

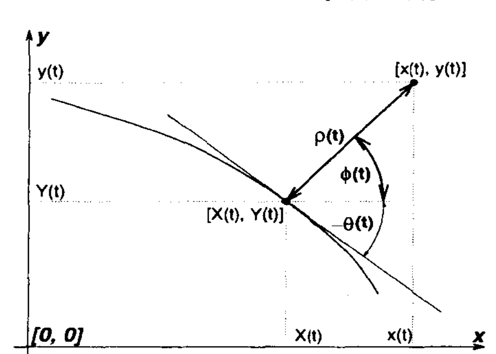
```
>> % main5.m
>> global a b m c w
>> a = 20; b = 15; % semi-axes
>> m = [10;20];
>> c = 17.58940018; % constant velocity of jogger
>> w = 19;  % velocity of dog
>> y0 = [60, 70, 0]';  % ini. cond., starting point of the dog

>> options= odeset('AbsTol',1e-5,'Events','on');
>> [t,Y]= ode23 ('fkt', [0 20], y0, options);
>> clf; hold on;
>> axis ([-10 70 -10 70]);
>> % axis ('equal');
>> title ('The Jogger Runs with Constant Velocity.');
>> p = length(t)-1;
>> for h=1:p
>>   plot ([Y(h,1), Y(h+1,1)] , [Y(h,2), Y(h+1,2)], '-', ...
>>        'Color', 'yellow', 'EraseMode','none');
>>   s1 = Y(h,3); s2 = Y(h+1,3);
>>   X1 = m(1) + a*cos(s1); Y1 = m(2) + b*sin(s1);
>>   X2 = m(1) + a*cos(s2); Y2 = m(2) + b*sin(s2);
>>   plot ([X1, X2] , [Y1, Y2], ':', ...
>>        'Color', 'green', 'EraseMode','none');
>>   drawnow;
>> end;
>> cross(Y(p,1),Y(p,2),2);
>> hold off;
```

1.7 利用移动的坐标系统

在本节中，我们将用 Cartesian 坐标系，分别描述小孩和慢跑者的位置，用活动的极坐标分别描述玩具和狗的位置，参见图 1.10, 活动的极坐标的原点，分别是小孩和慢跑者的位置，从而由距离 $\rho(t)$ 和极角 $\phi(t)$, 可确定玩具或狗的当前位置.

图 1.10 移动坐标系， $[\rho(t),\phi(t)]$



如果 $[X(t),Y(t)]$ 为当前小孩 / 玩具的位置，则在 Cartesian 坐标中，慢跑者 / 狗的坐标 $[x(t),y(t)]$ 是

$$x(t)=X(t)+\rho(t)\cos(\phi(t)),\ y(t)=Y(t)+\rho(t)\sin(\phi(t)). \tag{1.10}$$

我们用新变量 $\rho(t)$ 和 $\phi(t)$, 分别表示微分方程系统 (1.7) 和 (1.8), 由此可得微分方程的一个变换.

注意，这两个微分方程系统是

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix}=\frac{\sqrt{\dot{x}^2+\dot{y}^2}}{\left\|\begin{pmatrix}X-x\\ Y-y\end{pmatrix}\right\|}\begin{pmatrix}X-x\\ Y-y\end{pmatrix} \tag{1.11}$$

的特殊情况. 如果保持常数速度 $\sqrt{\dot{x}^2+\dot{y}^2}=w=const$, 则 (1.11) 描述慢跑者 / 狗问题. 另一方面，如果距离是常数

$$\rho(t)=\left\|\begin{pmatrix}X-x\\ Y-y\end{pmatrix}\right\|=a=const$$

则得到小孩 / 玩具问题的方程. 因为对于小孩 / 玩具问题， $\rho(t)=const$, 我们希望在极坐标中，简化成只有一个关于 $\phi(t)$ 的微分方程，我们将用 MAPLE 实现这种变换.

1.7.1 慢跑者 / 狗的变换

我们先定义系统 (1.8):

```
> restart;
> S(t) := sqrt((X(t) - x(t))^2 + (Y(t) - y(t))^2):
> rx := diff(x(t), t) = W*(X(t) - x(t))/S(t);
> ry := diff(y(t), t) = W*(Y(t) - y(t))/S(t);
```

$$rx:=\frac{\partial}{\partial t}\mathrm{x}(t)=\frac{\mathrm{W}\,(\mathrm{X}(t)-\mathrm{x}(t))}{\sqrt{(\mathrm{X}(t)-\mathrm{x}(t))^2+(\mathrm{Y}(t)-\mathrm{y}(t))^2}}$$

$$ry:=\frac{\partial}{\partial t}\mathrm{y}(t)=\frac{\mathrm{W}\,(\mathrm{Y}(t)-\mathrm{y}(t))}{\sqrt{(\mathrm{X}(t)-\mathrm{x}(t))^2+(\mathrm{Y}(t)-\mathrm{y}(t))^2}}$$

现在，我们引进函数的变换和它们的导数：

```
> Rx := x(t) = X(t) + rho(t)*cos(phi(t)):
> Ry := y(t) = Y(t) + rho(t)*sin(phi(t)):
> Vx := diff(Rx, t); Vy := diff(Ry, t);
```

$$Vx := \frac{\partial}{\partial t}\,\mathrm{x}(t) = (\frac{\partial}{\partial t}\,\mathrm{X}(t)) + (\frac{\partial}{\partial t}\,\rho(t))\cos(\phi(t)) - \rho(t)\sin(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t))$$

$$Vy := \frac{\partial}{\partial t}\,\mathrm{y}(t) = (\frac{\partial}{\partial t}\,\mathrm{Y}(t)) + (\frac{\partial}{\partial t}\,\rho(t))\sin(\phi(t)) + \rho(t)\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t))$$

我们代入并化简结果：

```
> qx := subs(Vx, rx):  qy := subs(Vy, ry):
> qx := simplify(subs(Rx, Ry, qx), symbolic);
> qy := simplify(subs(Rx, Ry, qy), symbolic);
```

$$qx := \quad (\tfrac{\partial}{\partial t}X(t)) + (\tfrac{\partial}{\partial t}\rho(t))\cos(\phi(t)) - \rho(t)\sin(\phi(t))(\tfrac{\partial}{\partial t}\phi(t)) = -W\cos(\phi(t))$$

$$qy := \quad \tfrac{\partial}{\partial t}Y(t) + (\tfrac{\partial}{\partial t}\rho(t))\sin(\phi(t)) + \rho(t)\cos(\phi(t))(\tfrac{\partial}{\partial t}\phi(t)) = -W\sin(\phi(t))$$

最后，为了得到 $\rho(t)$ 和 $\phi(t)$ 的导数，我们解：

```
> Dsys := solve({qx, qy}, {diff(rho(t), t), diff(phi(t), t)});
```

$$Dsys := \left\{\frac{\partial}{\partial t}\,\phi(t) = -\frac{\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\mathrm{Y}(t)) - (\frac{\partial}{\partial t}\,\mathrm{X}(t))\sin(\phi(t))}{\rho(t)},\right.$$
$$\left.\frac{\partial}{\partial t}\,\rho(t) = -(\frac{\partial}{\partial t}\,\mathrm{Y}(t))\sin(\phi(t)) - \cos(\phi(t))\,(\frac{\partial}{\partial t}\,\mathrm{X}(t)) - \mathrm{W}\right\}$$

```
> Dradial := Dsys[2]; Daxial := Dsys[1];
```

$$Dradial := \frac{\partial}{\partial t}\,\rho(t) = -(\frac{\partial}{\partial t}\,\mathrm{Y}(t))\sin(\phi(t)) - \cos(\phi(t))\,(\frac{\partial}{\partial t}\,\mathrm{X}(t)) - \mathrm{W} \tag{1.12}$$

$$Daxial := \frac{\partial}{\partial t}\,\phi(t) = -\frac{\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\mathrm{Y}(t)) - (\frac{\partial}{\partial t}\,\mathrm{X}(t))\sin(\phi(t))}{\rho(t)} \tag{1.13}$$

导出的微分方程系统更简单，但是没有解析解，因此我们将不继续讨论.

1.7.2 小孩 / 玩具的变换

如前所述，我们定义微分方程系统：

```
> Rx := x(t) = X(t) + a*cos(phi(t)):
> Ry := y(t) = Y(t) + a*sin(phi(t)):
> Vx := diff(Rx, t):  Vy := diff(Ry, t):
```

对于玩具的速度，我们将引进下列代入法 RW：

```
> RW := W(t) = subs(Vx, Vy, sqrt(diff(x(t), t)^2
>        + diff(y(t), t)^2));
```

$$RW := \mathrm{W}(t) = \sqrt{((\frac{\partial}{\partial t}\,\mathrm{X}(t)) - a\sin(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)))^2 + ((\frac{\partial}{\partial t}\,\mathrm{Y}(t)) + a\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)))^2}$$

下列语句生成系统 (1.7)

```
> qx := diff(x(t), t) = W(t)/a*(X(t) - x(t));
> qy := diff(y(t), t) = W(t)/a*(Y(t) - y(t));
```

$$qx := \frac{\partial}{\partial t}\,\mathrm{x}(t) = \frac{\mathrm{W}(t)\,(\mathrm{X}(t) - \mathrm{x}(t))}{a}$$

$$qy := \frac{\partial}{\partial t}\,\mathrm{y}(t) = \frac{\mathrm{W}(t)\,(\mathrm{Y}(t) - \mathrm{y}(t))}{a}$$

通过消去 $x(t)$ 和 $y(t)$, 我们引进新的变量和导数，并且我们对两个方程取平方，从而去掉平方根.

```
> qx := subs(RW, Vx, Rx, map(u -> u^2, qx));
> qy := subs(RW, Vy, Ry, map(u -> u^2, qy));
```

$$qx := ((\frac{\partial}{\partial t}\,\mathrm{X}(t)) - a\sin(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)))^2 =$$
$$(((\frac{\partial}{\partial t}\,\mathrm{X}(t)) - a\sin(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)))^2 + ((\frac{\partial}{\partial t}\,\mathrm{Y}(t)) + a\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)))^2)\cos(\phi(t))^2$$
$$qy := ((\frac{\partial}{\partial t}\,\mathrm{Y}(t)) + a\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)))^2 =$$
$$(((\frac{\partial}{\partial t}\,\mathrm{X}(t)) - a\sin(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)))^2 + ((\frac{\partial}{\partial t}\,\mathrm{Y}(t)) + a\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)))^2)\sin(\phi(t))^2$$

我们希望证明两个方程是相同的，这样系统将简化为一个关于 $\phi(t)$ 的微分方程，为此我们做下列的代入:

```
> Subst := [rhs(Vx) = A, rhs(Vy) = B, phi(t) = F];
```

$$Subst := [(\frac{\partial}{\partial t}\,\mathrm{X}(t)) - a\sin(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)) = A,\ (\frac{\partial}{\partial t}\,\mathrm{Y}(t)) + a\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)) = B,\ \phi(t) = F]$$

```
> q1x := expand(subs(Subst, qx));
> q1y := expand(subs(Subst, qy));
```

$$q1x := A^2 = \cos(F)^2\,A^2 + \cos(F)^2\,B^2$$
$$q1y := B^2 = \sin(F)^2\,A^2 + \sin(F)^2\,B^2$$

为了证明方程实际上是相同的，通过集中 A 和 B 项，简化方程.

```
> q1x := A^2*sin(F)^2 = B^2*cos(F)^2;
> q1y := B^2*cos(F)^2 = A^2*sin(F)^2;
```

$$q1x := \sin(F)^2\,A^2 = \cos(F)^2\,B^2$$
$$q1y := \cos(F)^2\,B^2 = \sin(F)^2\,A^2$$

因此它们是相同的. 我们用第一个进行计算，并去掉平方、代入. 通过求解 $\phi(t)$ 的导数，我们得到所求的微分方程:

```
> Q1 := map(u -> simplify(sqrt(u), symbolic), q1x);
```

$$Q1 := \sin(F)\,A = \cos(F)\,B$$

```
> BackSubst := [seq(rhs(op(i, Subst)) = lhs(op(i, Subst)), i = 1..nops(Subst))];
```

$$BackSubst := [A = (\frac{\partial}{\partial t}\,\mathrm{X}(t)) - a\sin(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)),\ B = (\frac{\partial}{\partial t}\,\mathrm{Y}(t)) + a\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)),\ F = \phi(t)]$$

```
> Q2 := subs(BackSubst, Q1);
```

$$Q2 := \sin(\phi(t))\,((\frac{\partial}{\partial t}\,\mathrm{X}(t)) - a\sin(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t))) = \cos(\phi(t))\,((\frac{\partial}{\partial t}\,\mathrm{Y}(t)) + a\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\phi(t)))$$

```
> Daxial := diff(phi(t), t) = simplify(solve(Q2, diff(phi(t), t)));
```

$$Daxial := \frac{\partial}{\partial t}\,\phi(t) = \frac{-\cos(\phi(t))\,(\frac{\partial}{\partial t}\,\mathrm{Y}(t)) + \sin(\phi(t))(\frac{\partial}{\partial t}\,\mathrm{X}(t))}{a}$$

值得注意，对于慢跑者 / 狗问题，在含有 $\phi(t)$ 的微分方程里，用常数 a 代替函数 $\rho(t)$，我们将得到相同的方程.

1.8 例子

由于微分方程系统已经简化成一个方程，所以对于某些情况，我们希望得到解析解. 首先，第一个例子是小孩在一个圆上行走.

```
> Child_Subst := X(t) = R*cos(omega*t), Y(t) = R*sin(omega*t);
```

$$Child_Subst := \mathrm{X}(t) = R\cos(\omega\,t),\ \mathrm{Y}(t) = R\sin(\omega\,t)$$

```
> Das := subs(Child_Subst, Daxial);
```

$$Das := \frac{\partial}{\partial t}\,\phi(t) = \frac{-\cos(\phi(t))\,(\frac{\partial}{\partial t}\,R\sin(\omega\,t)) + \sin(\phi(t))(\frac{\partial}{\partial t}\,R\cos(\omega\,t))}{a}$$

```
> Das := combine(Das, trig);
```

$$Das := \frac{\partial}{\partial t}\,\phi(t) = -\frac{R\,\omega\cos(-\phi(t) + \omega t)}{a}$$

```
> Sol := dsolve(Das, phi(t));
```

$$Sol := 2\,\frac{\arctan\left(\dfrac{(a-R)\tan(\frac{1}{2}\,\omega\,t - \frac{1}{2}\,\phi(t))}{\sqrt{(a+R)\,(a-R)}}\right)a}{\sqrt{a^2 - R^2\,\omega^2}} - \frac{t}{\omega} = _C1$$

我们得到一个解析的表达式！由一个隐式方程给出函数 $\phi(t)$. 因此，对于给定的初始条件， MAPLE 不能解此微分方程. 虽然在上面的方程里， solve 命令不能求解 $\phi(t)$, 但可用手工求解. 因为对于这种特殊情况，确定积分常数更为简单，所以在这里我们不解它.

为了重新给出图 1.3 中的轨迹，我们引入初始条件：

```
> IniSol := eval(subs(t = 0, phi(0) = alpha, Sol));
```

$$IniSol := -2\,\frac{\arctan\left(\dfrac{(a-R)\tan(\frac{1}{2}\,\alpha)}{\sqrt{(a+R)\,(a-R)}}\right)a}{\sqrt{a^2-R^2\,\omega^2}} = _C1$$

现在我们求解特解（利用 MAPLE ）：

```
> Sol := map(u -> (u + t/omega)*sqrt(a^2 - R^2)*omega^2/a/2, Sol):
> Sol := map(u -> tan(u)*sqrt((a + R)*(a - R))/(a - R), Sol):
> Sol := simplify(map(u -> arctan(u), Sol), symbolic):
> _C1 := lhs(IniSol):
> Sol := map(u -> (-2*u + omega*t), Sol):
```

对于图 1.3, 我们确定初始角 α：

```
> Sol0 := eval(subs(alpha = 0, Sol));
```

$$Sol0 := \phi(t) = -2\arctan\left(\frac{\tan(\frac{1}{2}\,\dfrac{t\,\omega\,\sqrt{a^2-R^2}}{a})\,\sqrt{(a+R)(a-R)}}{a-R}\right)+\omega\,t$$

我们希望令棒的长 a 等于圆的半径 R, 简单的替代是不可能的，我们必须计算 $a \to R$ 的极限：

```
> Sol00 := lhs(Sol0) = limit(rhs(Sol0), a = R, 'left'):
> Sol00 := simplify (Sol00, symbolic);
```

$$Sol00 := \phi(t) = -2\arctan(\omega\,t)+\omega\,t$$

```
> Toy := [x(t), y(t)]:
> Toy_Plot := subs(Rx, Ry, Child_Subst, Sol00, omega = 1,
>                  a = R, R = 5, [Toy[], t = 0..40]);
```

$$Toy_Plot := [5\cos(t)+5\cos(-2\arctan(t)+t),\ 5\sin(t)+5\sin(-2\arctan(t)+t),\ t=0..40]$$

```
> plot(Toy_Plot, scaling = constrained, color = black);
```

我们得到与图 1.3 同样的图形. 用下列语句可得图 1.5：

```
> Sol15 := eval(subs(alpha = arctan(-2)+Pi, Sol)):
> Toy_Plot := subs(Rx, Ry, Child_Subst, Sol15, omega = 1,
>                  a = sqrt(125), R = 5, [Toy[], t = 0..200]):
> plot(Toy_Plot, scaling = constrained, color = black);
```

最后，我们计算第一个例子中变量的轨迹，并考虑棒比圆的半径短的情况：

```
> Sol1 := evalc(Sol0):
> Sol1 := evalc(subs(signum(a^2 - R^2) = -1, Sol1)):
```

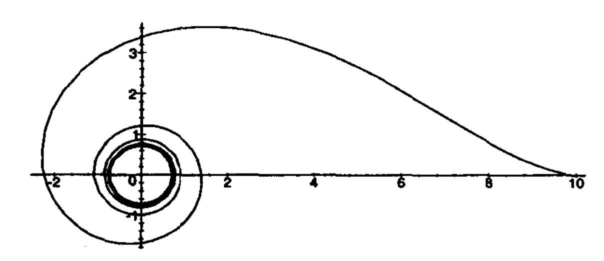
```
> Sol1 := subs(abs(a^2 - R^2) = R^2 - a^2, Sol1);
```

$$Sol1 := \phi(t) = 2\arctan\left(\frac{\sinh(\frac{1}{2}\frac{\omega t\sqrt{R^2-a^2}}{a})\cosh(\frac{1}{2}\frac{\omega t\sqrt{R^2-a^2}}{a})\sqrt{R^2-a^2}}{(1+\sinh(\frac{1}{2}\frac{\omega t\sqrt{R^2-a^2}}{a})^2)(a-R)}\right)+\omega t$$

```
> Toy_Plot := subs(Rx, Ry, Child_Subst, Sol1, omega = 1,
>                  a = 4.95, R = 5, [Toy[], t = 0..40]):
> plot(Toy_Plot, scaling = constrained, color = black);
```

图 1.11 给出此轨迹.

图 1.11 $a < R$ 时玩具的轨道



注意，如果小孩在直线上行走，则有一个解析解：等切面曲线，如本章第一节所示．令 (X_0, Y_0) 是小孩的初始位置， (V_x, V_y) 是他的恒定速度向量：

```
> a := 'a':  Vx := 'Vx':  Vy := 'Vy':
> Child_Subst := X(t) = Xo + Vx*t, Y(t) = Yo + Vy*t:
> Das := combine(subs(Child_Subst, Daxial));
```

$$Das := \frac{\partial}{\partial t}\phi(t) = \frac{-\cos(\phi(t))\,Vy + \sin(\phi(t))\,Vx}{a}$$

```
> Sol := dsolve(Das, phi(t));
```

$$Sol := 2\,\frac{\operatorname{arctanh}\left(\frac{1}{2}\,\frac{2\,Vy\tan(\frac{1}{2}\phi(t)) + 2\,Vx}{\sqrt{Vy^2+Vx^2}}\right)a}{\sqrt{Vy^2+Vx^2}} + t = _C2$$

重复前面的小孩在圆上行走的情况，我们又得到 $\phi(t)$ 的一个隐函数.

总之，对于玩具的轨迹，不太可能得到一个解析解．如果考虑小孩在正弦曲线上行走，则只能数值地计算玩具的轨迹.

```
> X := t -> t:  Y := t -> 5*sin(t):  a:= 10:
> Daxial;
```

$$\frac{\partial}{\partial t}\phi(t) = -\frac{1}{2}\cos(\phi(t))\cos(t) + \frac{1}{10}\sin(\phi(t))$$

```
> F := dsolve({Daxial, phi(0) = Pi/2}, phi(t), numeric);
```

$$F := \mathbf{proc}(rkf45_x)\ \ldots\ \mathbf{end}$$

因为此图形与图 1.4 相同，所以此处没有给出．然而我们将增加一些命令，以便在屏幕上动态地显示这些运动．注意，这里还无法用 **animate** 命令．

我们想看棒的运动，以及它两端的运动轨迹．一端必须沿正弦曲线移动，另一端将描述所计算的轨迹．我们希望看到，在 T 秒内绘出 N 个点序列过程的片段．

```
> T := 6*Pi:  N := 200:  L := a:
> SF := [evalf(seq(rhs(F(T*i/N)[2]), i = 0..N))]:
> ST := [evalf(seq(T*i/N, i= 0..N))]:
> plot([seq([X(ST[i]) + L*cos(SF[i]), Y(ST[i]) + L*sin(SF[i])], i =1..N)]);
```

上面最后的 MAPLE 命令将显示此轨迹．为了此片段，我们必须准备和存储绘图点列，该点列是由递增数给出的．

```
> TS := [seq(op(1, plot([seq([X(ST[i]) + L*cos(SF[i]),
>       Y(ST[i]) + L*sin(SF[i])], i = 1..j)])), j = 1..N)]:
> BS := [seq(op(1, plot([[X(ST[i]), Y(ST[i])],
>       [X(ST[i]) + L*cos(SF[i]), Y(ST[i]) + L*sin(SF[i])]])),
>       i=1..N)]:
> PS := [seq(op(1, plot([seq([X(ST[i]), Y(ST[i])],
>       i = 1..j)])), j = 1..N)]:
```

序列 TS 包含了轨迹的信息，它存储从 1 到 N 的 N 个部分图形点．序列 BS 描述棒的位置．PS 描述正弦轨迹，它的结构类似于 TS.

现在用命令

```
> PLOT(ANIMATE(seq([TS[i], BS[i], PS[i]], i=1..N)),
>       AXESLABELS(x, y), VIEW(-2..X(ST[N]), DEFAULT));
```

可看到整个过程．

参考文献

[1] E. HAIRER, S.P. NØRSETT AND G. WANNER, *Solving Ordinary Differential Equations I*, Springer-Verlag Berlin Heidelberg, 1987.

[2] H. HEUSER, *Gewöhnliche Differentialgleichungen*, B. G. Teubner, Stuttgart, 1989.

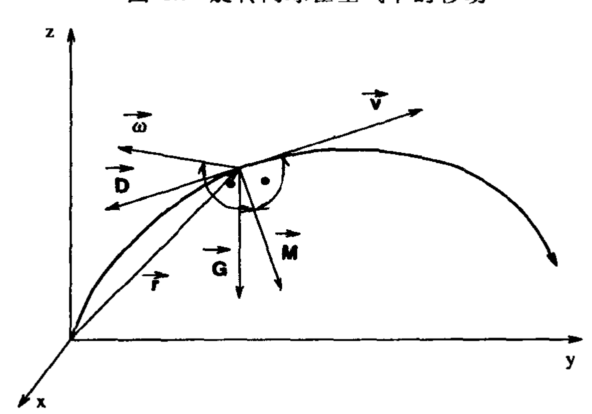
第二章　旋转网球的轨迹

F. Klvaňa

2.1　引言

考虑一个质量为 m, 直径为 d 的网球, 在靠近地球表面的空中运动. 此球以角速度 $\vec{\omega}$ 旋转 (向量 $\vec{\omega}$ 有旋转轴的方向和大小 $\omega = \mathrm{d}\varphi(t)/\mathrm{d}t = \dot{\varphi}(t)$, 其中 $\varphi(t)$ 是旋转的角度). 我们将笛卡儿坐标系 (xyz) 放在地球的表面, 其中 z 轴为垂直方向.

图 2.1　旋转网球在空气中的移动



我们把网球看成一个质点, 它在下列一些力的作用下运动 (参见图 2.1)

- 重力 $\vec{G} = m\,\vec{g}$, 其中 $\vec{g} = (0, 0, -g)$ 是重力加速度向量.
- 拉力 $\vec{D} = -D_L(v)\,\vec{v}\,/v$, 它与速度 $\vec{v}$ 的方向相反.
- Magnus 力 $\vec{M} = M_L\,\vec{\omega}\,/\omega \times \vec{v}\,/v$; 此力垂直于 $\vec{v}$ 和 $\vec{\omega}$.

拉力的大小 $D_L(v)$ 和 Magnus 力的大小 $M_L(v)$, 通常假设由理想的流体理论 [1] 给出:

$$\begin{aligned} D_L(v) &= C_D \frac{1}{2} \frac{\pi d^2}{4} \rho v^2 \\ M_L(v) &= C_M \frac{1}{2} \frac{\pi d^2}{4} \rho v^2 \end{aligned} \tag{2.1}$$

其中 ρ 是空气密度. 对于实际的流体 (空气), 系数 C_D 和 C_M 依赖于速度 v, 球的旋转和它的表面材料. 通常我们由实验得到这些系数.

[2] 报告了旋转网球的实验结果. 实验结果表明, 对于一个速度 $v \in [13.6, 28][ms^{-1}]$ 和球的转数 $n \in [800, 3250]rpm$ 的网球, 系数 C_D 和 C_M 只依赖于 v/w, 其中 $w = d/2$. 在某种意义上,

$|\vec{\omega} \times \vec{v}/v|$ 是转球赤道的速度 $\omega d/2$ 在速度向量 $\vec{v}$ 上的投影. 下面是系数的表达式:

$$\begin{aligned} C_D &= 0.508 + \left(\frac{1}{22.053 + 4.196(\frac{v}{w})^{5/2}}\right)^{2/5} \\ C_M &= \frac{1}{2.022 + 0.981(\frac{v}{w})} \end{aligned} \tag{2.2}$$

对于网球, 我们可以忽略球旋转的减速度, 因此 w 是常数.

于是, 由位置向量 $\vec{r}(t)$ 的 Newton 方程, 可定义球的轨道

$$m\frac{\mathrm{d}^2 \vec{r}(t)}{\mathrm{d}t^2} = -m\vec{g} - D_L\frac{\vec{v}}{v} + M_L\frac{\vec{\omega}}{\omega} \times \frac{\vec{v}}{v} \tag{2.3}$$

其中初始条件为

$$\vec{r}(0) = \vec{r}_0 \text{ 和 } \frac{\mathrm{d}\vec{r}}{\mathrm{d}t}(0) = \vec{v}_0 .$$

方程 (2.3) 是三个微分方程的非线性系统, 不存在解析解, 因此我们必须数值地求解.

实际上, 大多数情况是高速旋转的高球, 角速度向量位于在水平面内, 并且垂直于向量 $\vec{v}_0$, 由方程 (2.3), 对于 $t \geq 0$, 它也垂直于 $\vec{v}(t)$, 因此轨道在垂直平面内. 选取 x 轴在此平面内, 于是方程 (2.3) 的最后形式是

$$\begin{aligned} \ddot{x} &= -C_D\,\alpha\,v\,\dot{x} + \eta C_M\,\alpha\,v\,\dot{z} \\ \ddot{z} &= -g - C_D\,\alpha\,v\,\dot{z} - \eta C_M\,\alpha\,v\,\dot{x} \end{aligned} \tag{2.4}$$

其中 $v = \sqrt{\dot{x}^2 + \dot{z}^2}$ 和 $\alpha = (\rho\pi d^2)/(8m)$. 参数 $\eta = \pm 1$ 描述旋转的方向 (对于高速旋转, $\eta = 1$).

对于 $t = 0$, 我们取初始条件

$$x(0) = 0,\ z(0) = h,\ \dot{x}(0) = v_0\cos(\vartheta),\ \dot{z}(0) = v_0\sin(\vartheta) \tag{2.5}$$

其中 v_0 是初始速度向量 $\vec{v_0}$ 的大小, ϑ 是 $\vec{v_0}$ 与 x 轴之间的夹角.

2.2 Maple 解法

为了说明拉力和 Magnus 力对球的轨迹的影响, 我们将考虑三种模型: 在真空中的球, 在空气中没有旋转的球, 和在空气中旋转的球.

在真空中的球

在真空中, 只有重力作用在球上, 方程 (2.4) 有一个非常简单的形式

$$\ddot{x} = 0,\ \ \ddot{z} = -g. \tag{2.6}$$

Maple 用函数 dsolve 求解 (2.6) 的通解:

```
>                # motion of ball in vacuum
> eqnid := diff(x(t), t$2) = 0, diff(z(t),t$2) = -g:
> varid := {x(t), z(t)}:
> initcid := x(0) = 0, z(0) = h,
>              D(x)(0) = v0*cos(theta), D(z)(0) = v0*sin(theta):
```

```
>                #solution for ball in vacuum
> resid := dsolve({eqnid, initcid}, varid);
```

$$resid := \{\mathrm{x}(t) = t\, v0 \cos(\theta),\ \mathrm{z}(t) = h + t\, v0 \sin(\theta) - \frac{1}{2}\, t^2\, g\}$$

注意，函数 dsolve 以方程组的形式

$$\langle varname \rangle = \langle expression \rangle$$

返回解 $\langle result \rangle$, 因此没有定义这些方程的顺序. 为了访问对应 $\langle varname \rangle$ 的解，我们将用函数

$$subs(\langle result \rangle, \langle varname \rangle).$$

在空气中的球

对于在空气中的球, 我们不能解析地求解方程 (2.4), 必须用数值解法. 因为后面要用到 Newton 法计算飞行时间，我们需要 $z(t)$ 的导数，因此将二阶微分方程组 (2.4) 转变成一阶微分方程组，(用变量 v_x 和 v_z 分别表示速度 $\dot{x}$ 和 $\dot{z}$),

$$\begin{aligned}
\dot{x} &= v_x \\
\dot{v}_x &= -C_L \alpha v \cdot v_x + \eta C_M \alpha v \cdot v_z \\
\dot{z} &= v_z \\
\dot{v}_z &= -g - C_L \alpha v \cdot v_z - \eta C_M \alpha v \cdot v_x
\end{aligned} \tag{2.7}$$

其中 $v = \sqrt{v_x^2 + v_z^2}$. 初始条件 (2.5) 为

$$x(0) = 0,\ z(0) = h,\ v_x(0) = v_0 \cos(\vartheta),\ v_z(0) = v_0 \sin(\vartheta). \tag{2.8}$$

我们将用 SI 单位系统，参数的数值是 $g = 9.81[ms^{-1}]$, 网球的直径 $d = 0.063[m]$, 其质量 $m = 0.05[kg]$, 空气密度 $\rho = 1.29[kgm^{-3}]$, 我们选取初始条件

$$h = 1\ [m],\ v_0 = 25\ [ms^{-1}],\ \vartheta = 15^\circ$$

和球旋转的参数 $w = 20\ [ms^{-1}]$ 和 $\eta = 1$(高速旋转), 用 MAPLE 语句求解此问题 [1] 如下所示.

```
>                # numeric solution of ball in air
>                # Basic constants in SI units
> g := 9.81:  # gravitational acceleration:
> d := 0.063:  m := 0.05:  # diameter and mass of ball
> rov := 1.29:  # density of air
> alpha := evalf(Pi*d^2/(8*m)*rov):
> v := (dx^2 + dz^2)^(1/2):  # definition of velocity
> Cd := .508+1/(22.503 + 4.196/(w/v(t))^(5/2))^(2/5):
> Cm := eta/(2.202 + .981/(w/v(t))):
>                # eta =+-1 defines direction of rotation,
>                # for top spinning eta = 1
> var := {x(t), z(t), dx(t), dz(t)}:
>                # initial conditions
```

[1] 此例子已用 MAPLE V Release 3 测试过

```
> initc := x(0) = 0, z(0) = h,
>          dx(0) = v0*cos(theta), dz(0) = v0*sin(theta):
>              # equations of motion of ball in air
>              # rotation (drag force only)
> eqnt0:= diff(x(t),t) = dx(t),
>         diff(dx(t),t)= -0.508*alpha*dx(t)*v(t),
>         diff(z(t),t) = dz(t),
>         diff(dz(t),t)= -g-0.508*alpha*dz(t)*v(t):
>              # equations of motion of rotating ball in air
>              # (influence of Magnus effect)
> eqnt1:= diff(x(t),t) = dx(t),
>         diff(dx(t),t)= (-Cd*dx(t) + Cm*dz(t))*alpha*v(t),
>         diff(z(t),t) = dz(t),
>         diff(dz(t),t)= -g-(Cd*dz(t) + Cm*dx(t))*alpha*v(t):
>              #numeric values of initial parameters
> h := 1:  v0 := 25:  theta := Pi/180*15:  # theta= 15 degrees
> w := 20:  eta := 1:
>              # solution for non rotating ball
> res0 := dsolve({eqnt0,initc},var,numeric);
```

$$res0 := \textbf{proc}(rkf45_x)\ \ldots\ \textbf{end}$$

```
>              # result is a set of equations
> res0(0.5);
```

$$[t = .5,\ \mathrm{x}(t) = 10.76902243898168,\ \mathrm{z}(t) = 2.745476721738618,$$
$$\mathrm{dx}(t) = 19.31511754217100,\ \mathrm{dz}(t) = .7584804887012169]$$

```
>              # solution for rotating ball
> res1 := dsolve({eqnt1, initc}, var, numeric);
```

$$res1 := \textbf{proc}(rkf45_x)\ \ldots\ \textbf{end}$$

函数 dsolve(..., numeric) 返回带一个参数 (独立变量) 的函数. 调用此函数得到一组方程

$$\langle variable\rangle = \langle value\rangle$$

(在此情况下, 变量是 $t, x(t), z(t), v_x(t), v_z(t)$). 为了访问数值, 我们仍用函数 subs.

绘制图形

作为解法的最后部分, 我们将在同一图中, 画出三个模型的轨道. 为此我们必须解一个求飞行时间 (方程 $z(t) = 0$ 的解) 的子问题.

为了计算真空中球的飞行时间 t_{maxid}, 我们用 (2.6) 的符号解 res_{id} 和函数 fsolve.

```
>              # calculation of tmax - time of falling to earth
>              # for ball in vacuum
> tmaxid := fsolve(subs(resid, z(t)) = 0, t, t = 0..5);
```

$$tmaxid := 1.458903640$$

在其它情况下，因为通过对微分方程积分，可得到 z 的导数 v_z，所以我们容易用 Newton 法求解方程 $z(t)=0$. 于是方程 $z(t)=0$ 的近似解的递推关系是

$$t_{n+1}=t_n-\frac{z(t_n)}{v_z(t_n)}. \tag{2.9}$$

算法 2.1 函数 zzero

```
zzero := proc (u, t0, z, dz) local tn, ts, up;
        # find root of z(t) = subs(u(t), z) = 0
        # using Newton method
        # using diff(z, t) = subs(u(t), dz)
   tn := t0; ts := 0.0;
   while abs((tn - ts)/tn) > 10^(-4) do;
         ts := tn;
         up := u(ts);
         tn := ts - subs(up, z)/subs(up, dz);
   od;
   tn;
end;
```

我们可用 t_{maxid} 作为初始近似解 t_0. MAPLE 函数 zzero 实现此 Newton 迭代 (参见算法 2.1).

```
>          # calculation of the flight time for the other models
> tmax0:= zzero(res0, tmaxid, z(t), dz(t));
> tmax1:= zzero(res1, tmaxid, z(t), dz(t));
```

$$tmax0 := 1.362013689$$
$$tmax1 := .9472277855$$

为了从数值解生成空气中球的轨道，最简单 (可能最快) 的方法是用一个数组 $[x(t_i), z(t_i)]$ 作为函数 plot 的输入参数. 对于一个具有常数时间步长的区间，为了从 dsolve(..., numeric) 的结果得到此数组，我们定义简单函数 *tablepar*. 为了产生红，蓝，黑色的图形，剩余的 MAPLE 语句是

```
>          # making graphs:
> Gid := plot([subs(resid, x(t)), subs(resid, z(t)),
>                t = 0..tmaxid], linestyle = 2):
>          # for models with numeric solution
>          # calculation of tables [x(t),z(t)] for plotting
> tablepar := proc(u, x, y, xmin, xmax, npoints) local i,Step;
>     Step := (xmax - xmin)/npoints;
>     [seq([subs(u(xmin + i*Step), x), subs(u(xmin + i*Step) ,y)],
>          i = 0 ..  npoints)]
> end:
> G0 := plot(tablepar(res0, x(t), z(t), 0, tmax0, 15),
>              linestyle = 3):
> G1 := plot(tablepar(res1, x(t), z(t), 0, tmax1, 15),
```

```
>             linestyle = 1):
>          # plotting of all graphs
>  plots[display]({Gid, G0, G1});
```

用 MATLAB 计算轨道，可得到相同的这些图形 (参见图 2.2).

算法 2.2 必要的 M 函数集合

```
function xdot= tennisip(t,x,flag)
  global g
  xdot = [ x(3)
           x(4)
           0
           -g  ];
function xdot= tennis0p(t,x)
  global g alpha
  v= sqrt(x(3)^2+x(4)^2);
  xdot= [ x(3)
          x(4)
          -alpha*0.508*x(3)*v
          -g-alpha*0.508*x(4)*v ];

function xdot= tennis1p(t,x)
  global g alpha w etha
  v = sqrt(x(3)^2 + x(4)^2);
  Cd = (0.508 + 1/(22.503 + 4.196*(v/w)^0.4))*alpha*v;
  Cm = etha*w/(2.022*w + 0.981*v)*alpha*v;
  xdot = [ x(3)
           x(4)
           -Cd*x(3) + Cm*x(4)
           -g-Cd*x(4) - Cm*x(3) ];
```

2.3 MATLAB 解法

由于问题是非线性的，我们必须用数值方法求解. 象 MATLAB 这样的数值系统比符号系统更合适. 因此我们将用 MATLAB 求解问题.

为了求解 n 个微分方程的系统的初值问题

$$\frac{\mathrm{d}\vec{x}}{\mathrm{d}t}=\vec{f}(t,\vec{x}), \tag{2.10}$$

其中 $\vec{x}=(x_1,x_2,\cdots,x_n)$, 我们用 MATLAB 函数 ode23(或 ode45), 实现 2 阶和 3 阶 (或 4 阶和 5 阶) 的嵌入式 Runge-Kutta 方法.

我们必须对每个模型定义 M 函数，以实现微分方程系统 (2.10). 为了在 M 文件之间传递参数 g, α, w 和 η, 在我们的程序中，定义它们为全局变量. 用如下映射

$$x \to x(1),\ z \to x(2),\ v_x \to x(3),\ v_z \to x(4).$$

所需的 M 文件在算法 (2.2) 中.

在下面主程序中，我们计算和绘制所有三个模型的轨道 (参见图 2.2). 为了画出球在空气中的轨道，计算一个充分密集的表 $[x_i, z_i]$, 我们用 $z = z(x)$ 的 100 个解点作样条函数插值，所用函数为 spline.

```
>>     % Trajectory of spinning tennis ball
>>     % initialization
>> global g alpha w eta
>>     % basic constants in MKS units
>> g = 9.81; d = 0.063; m = 0.05; rho = 1.29
>> alpha = pi*d^2/(8*m)*rho;
>> eta = 1;
>> w = 20;
>>     % initial conditions
>> h = 1; v0 = 25; theta = pi/180*15;
>> xin = [0, h, v0*cos(theta), v0*sin(theta)];
>>     % flight time for vacuum
>> tmaxid = (xin(4) + sqrt(xin(4)^2 + 2*g*xin(2)))/g;
>>     % solution in vacuum
>> [tid, xid] = ode23('tennisip', [0 tmaxid], xin);
>>     % solution without spin
>> [t0, x0] = ode23('tennis0p', [0 tmaxid], xin);
>>     % solution with spin
>> [t1, x1] = ode23('tennis1p', [0 tmaxid], xin);
>> N = max(xid(:,1)); x = 0:N/100:N;
>> axis([0, max(xid(:,1)), 0 , max(xid(:,2))])
>> hold on;
>> plot(x, spline(xid(:,1), xid(:,2), x), ':r');
>> plot(x, spline(x0(:,1), x0(:,2), x), '--b');
>> plot(x, spline(x1(:,1), x1(:,2), x), '-w');
>> hold off;
```

注意，在空气中的两个模型，我们不必计算飞行时间. 对于最大的轨道 (在真空中的球), 利用紧跟 axis 语句后的 hold 语句，轨道被截断，并且不画出 x 轴以下的图形.

如果希望计算飞行时间，类似于 MAPLE 解法，我们可以求解. 因为从开始就必须对微分方程组数值积分，得到函数值和导数，这种解法要花费很多的计算时间.

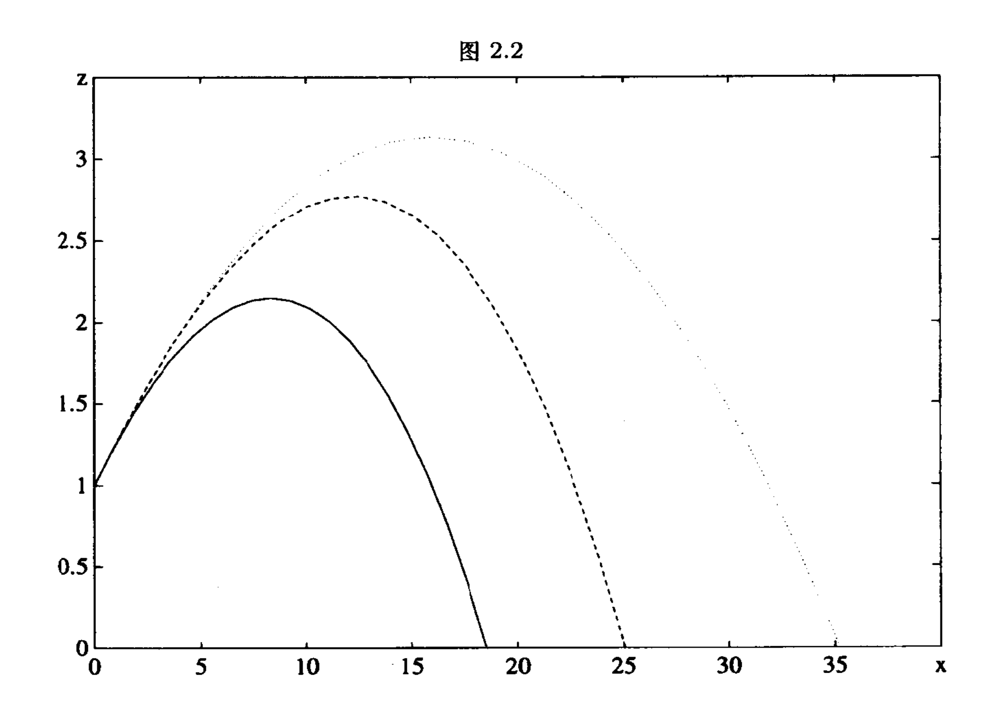
然而，用逆插法计算飞行时间的近似值更简单. 为此我们要找一个时间区间，在此区间上函数是可逆的. 我们取表中的某些值，在这些值处，轨道的 z 值是单调递减的. 这可由函数 min 和

max 完成:

```
>>      % Determine flight time by inverse interpolation
>> [z,j]=min(x0(:,2));  [y,i]=max(x0(:,2));
>> tmax0 = spline(x0(i+1:j,2), t0(i+1:j), 0)
>> [z,j]=min(x1(:,2));  [y,i]=max(x1(:,2));
>> tmax1 = spline(x1(i+1:j,2), t1(i+1:j), 0)
```

为了使区域 $z(i+1:j)$ 是单调的, 我们取第 $(i+1)$ 个点为 z 的单调区域的第一个点. 于是得到飞行时间的近似值 $t_{max0} = 1.3620$ 和 $t_{max1} = 0.9193$, 比 MAPLE 的结果好.

图 2.2



2.4 MATLAB 5 更简单的解法

MATLAB 5 允许对微分方程求积, 直到所得的轨道交于 x 轴 [2], 因此不需要样条插值. 根据第一章第 7 页的解释, 为了检查零交点, 必须调整算法 (2.2) 中定义的 M 函数; 算法 (2.3) 给出修改的 M 函数. 计算三个轨迹的主程序变得更短: 选项 Events 设置成 on, 使积分器检查 z 轴的零交点, 如果有一个, 则停止积分. 因此可由积分器返回的时间向量 t_0 和 t_1 的最后部分, 得到轨迹的时间.

算法 2.3 在 MATLAB 5 下重写上一节的 M 文件

```
function [xdot,isterminal,direction]= tennisip2(t,x,flag)
global g
if nargin < 3 | isempty(flag) % normal output
```

[2]Leonhard Jaschke 给出此解

```
    xdot = [x(3), x(4), 0, -g]';
else
    switch(flag)
    case 'events'  % at ||h||=0 there is a singularity
        xdot= x(2);
        isterminal= 1;
        direction= -1;
    otherwise
        error(['Unknown flag ''' flag '''.']);
    end
end

function [xdot,isterminal,direction]= tennis0p2(t,x,flag)
global g alpha
if nargin < 3 | isempty(flag) % normal output
    v= sqrt(x(3)^2+x(4)^2);
    xdot = [x(3), x(4), ...
            -alpha*0.508*x(3)*v, -g-alpha*0.508*x(4)*v]';
else
    switch(flag)
    case 'events'  % at ||h||=0 there is a singularity
        xdot= x(2);
        isterminal= 1;
        direction= -1;
    otherwise
        error(['Unknown flag ''' flag '''.']);
    end
end

function [xdot,isterminal,direction]= tennis1p2(t,x,flag)
global g alpha w etha
if nargin < 3 | isempty(flag) % normal output
    v= sqrt(x(3)^2+x(4)^2);
    Cd=(0.508+1/(22.503+4.196*(v/w)^0.4))*alpha*v;
    Cm=etha*w/(2.022*w+0.981*v)*alpha*v;
    xdot = [x(3), x(4), ...
            -Cd*x(3)+Cm*x(4), -g-Cd*x(4)-Cm*x(3)]';
else
    switch(flag)
    case 'events'  % at ||h||=0 there is a singularity
```

```
          xdot= x(2);
          isterminal= 1;
          direction= -1;
       otherwise
          error(['Unknown flag ''' flag '''.']);
       end
    end
```

积分器取相当大的积分步长，使得轨迹的图形没有样条插值所得的光滑．为了使这些轨迹光滑，我们必须取小步长．用选项 Maxstep 规定步长的上界来实现．通常设此值为时间区间长的十分之一．在下列算法中，我们令它为 tmaxid/100, 可得到如图 2.2 所示的同样光滑的轨迹：

```
>>      % Trajectory of rotating tennis ball
>>      % initialization
>> global g alpha w etha
>> g = 9.81; d = 0.063; m = 0.05; rho = 1.29;
>> alpha = pi*d^2/(8*m)*rho;
>> etha = 1;
>> w = 20;
>>      % initial conditions
>> h = 1; v0 = 25; theta = pi/180*15;
>> xin = [0, h, v0*cos(theta), v0*sin(theta)]
>>      % flight time for vacuum
>> tmaxid = (xin(4)+sqrt(xin(4)^2+2*g*xin(2)))/g
>>      % setting options for the integrator
>> options=odeset('Events','on','Maxstep',tmaxid/100);
>>      % solution in vacuum
>> [tid, xid]=ode23('tennisip2',[0 tmaxid],xin,options);
>>      % solution without spin
>> [t0, x0] = ode23('tennis0p2',[0 tmaxid],xin,options);
>>      % solution with spin
>> [t1, x1] = ode23('tennis1p2',[0 tmaxid],xin,options);
>> clf;
>> axis([0,max(xid(:,1))*1.13,0,max(xid(:,2))*1.13]);
>> hold on;
>> plot(xid(:,1), xid(:,2), ':r');
>> plot(x0(:,1), x0(:,2), '--b');
>> plot(x1(:,1), x1(:,2), '-g');
>> hold off;
>>      % flight time for the trajectory without spin
>> tmax0 = t0(length(t0))
>>      % flight time for the trajectory with spin
```

```
>> tmax1 = t1(length(t1))
```

参考文献

[1] E.G. RICHARDSON, *Dynamics of Real Fluids*, Edward Arnold, 1961.

[2] A. ŠTĚPÁNEK, *The Aerodynamics of Tennis Balls - The Topspin Lob*, American Journal of Physics, 56,1988, pp. 138-142.

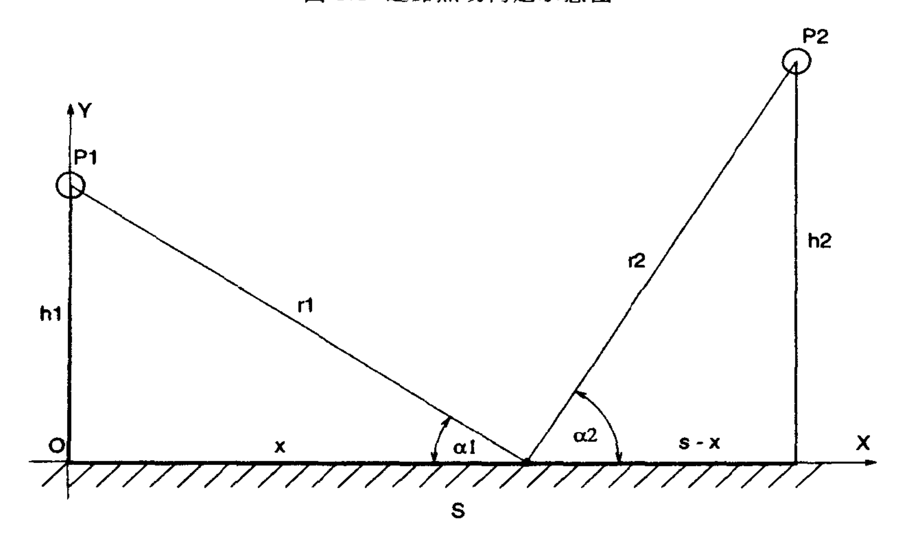
第三章　道路照明问题

S.Bartoň, D.Gruntz

3.1　引言

这一章我们研究由两盏路灯照明的一条水平的道路，其中 P_i 是路灯的亮度， h_i 是灯的高度. 两盏路灯的坐标分别是 $(0,h_1)$ 和 (s,h_2) ，其中 s 是两灯之间的水平距离. 令 $X=(x,0)$ 是两灯之间道路上的一个点. 这一章我们将找出具有最小照明强度的点 X. 图 3.1 给出了在这一章将要讨论的问题的一个示意图.

图 3.1　道路照明问题示意图



在第一节我们将在给定两灯的高度和亮度的前提下求出 X 点的位置. 第二节我们通过改变第二盏灯的高度以极大化 X 点的照明强度. 在最后一节我们还将进一步优化与两灯高度有关的 X 点的照明问题. 事实上，我们将通过改变两灯的高度来优化道路照明问题.

3.2　道路上的最低照明强度的点

这一节我们将寻找两个光源之间照明强度最低的点 X. 由物理学的知识可知：被光线照射的物体的亮度依赖于它与光源之间的距离平方的倒数和光线的投射角度，参见 [1,2].

从 X 到第一个光源的水平距离是 x ，到第二个的水平距离是 $s-x$. 利用勾股定理我们可以确定点 X 到两个光源之间的距离 r_1 和 r_2 ，

$$r_1^2=h_1^2+x^2,\quad r_2^2=h_2^2+(s-x)^2.$$

从两盏灯到 X 点的光线强度分别为

$$I_1(x)=\frac{P_1}{r_1^2}=\frac{P_1}{h_1^2+x^2},\quad I_2(x)=\frac{P_2}{r_2^2}=\frac{P_2}{h_2^2+(s-x)^2}.$$

如果光线的投射角度分别为 α_1 和 α_2, 道路的照明情况还依赖于 $\sin\alpha_1$ 和 $\sin\alpha_2$

$$\sin\alpha_1=\frac{h_1}{\sqrt{h_1^2+x^2}},\quad \sin\alpha_2=\frac{h_2}{\sqrt{h_2^2+(s-x)^2}}.$$

于是 X 点的总的照明强度 $C(x)$ 是

$$C(x)=I_1(x)\sin\alpha_1+I_2(x)\sin\alpha_2=\frac{P_1h_1}{\sqrt{(h_1^2+x^2)^3}}+\frac{P_2h_2}{\sqrt{\left(h_2^2+(s-x)^2\right)^3}}. \tag{3.1}$$

对于 $0\le x\le s$, 极小化 $C(x)$ 就得到 X 点的坐标. 为了求出极小值, 可以对 $C(x)$ 求导并求出它的根. 我们尝试使用 MAPLE 来做此事.

```
> S[1] := P[1]*h[1]/(h[1]^2+x^2)^(3/2):
> S[2] := P[2]*h[2]/(h[2]^2+(s-x)^2)^(3/2):
> C := S[1]+S[2]:
> dC :=diff(C,x);
```

$$dC:=-3\,\frac{P_1\,h_1\,x}{({h_1}^2+x^2)^{5/2}}-\frac{3}{2}\,\frac{P_2\,h_2\,(-2\,s+2\,x)}{({h_2}^2+(s-x)^2)^{5/2}}$$

```
> solve(dC=0,x):
```

如果你试图执行这个指令, 你将会看到 MAPLE 决不会给你回答. 使用代数学的处理方法, 方程 $dC=0$ 能够被转换为关于 x 的多项式方程. 我们把方程 $dC=0$ 的一项移到方程的右端, 两边平方, 再把右端移回左端, 通分. 则分子必须等于零.

```
> eq := diff(S[1], x)^2 - diff(S[2], x)^2;
```

$$eq:=9\,\frac{{P_1}^2\,{h_1}^2\,x^2}{({h_1}^2+x^2)^5}-\frac{9}{4}\,\frac{{P_2}^2\,{h_2}^2\,(-2\,s+2\,x)^2}{({h_2}^2+(s-x)^2)^5}$$

```
> eq := collect(primpart(numer(eq)),x);
```

最后一个指令的结果是一个关于 x 的 12 次的多项式,

$$(P_1^2h_1^2-P_2^2h_2^2)x^{12}+(2P_2^2h_2^2s-10P_1^2h_1^2s)x^{11}+\ldots-P_2^2h_2^2h_1^{10}s^2=0. \tag{3.2}$$

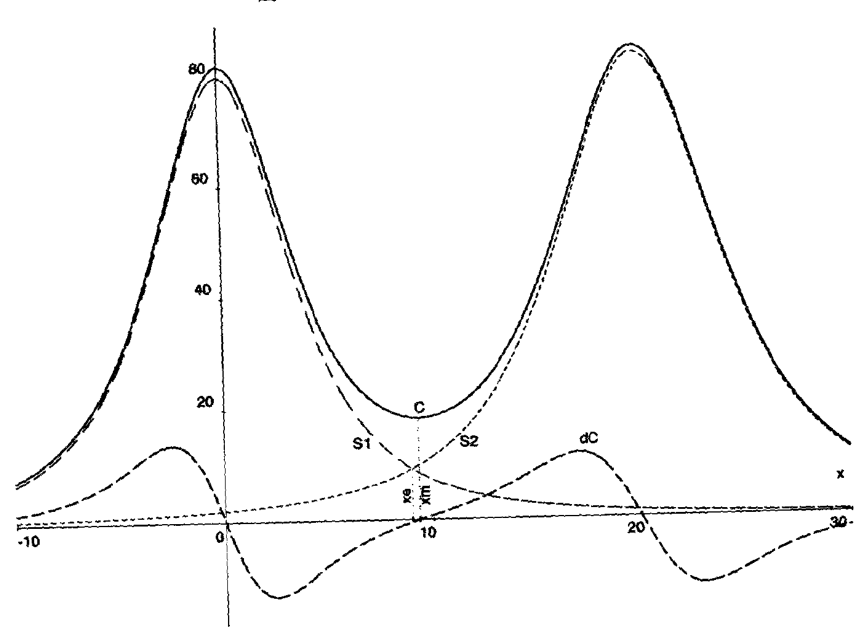
式中的常数不确定时, 这个多项式很难解出其封闭的形式.

我们考虑如下的数值: $P_1=2000[W], P_2=3000[W], h_1=5[m], h_2=6[m], s=20[m]$. 函数 $C(x), C'(x)\equiv dC, S_1(x)$ 和 $S_2(x)$(每一盏灯分别投射到道路上的照明强度) 可以使用 MAPLE 在图上绘制出来 (见图 3.2). 从图上可以辨别出包含函数 $C'(x)$ 的零点的区间. 我们将在这个区间上用 `fsolve` 求出 X 的数值.

```
> P[1] := 2000:  P[2] := 3000:  s := 20:  h[1] := 5:  h[2]:=6:
> plot({C, S[1], S[2], dC}, x = -s/2..s*3/2);
> Xm := fsolve(dC=0,x, 5..10);
```

$$Xm:=9.338299136$$

图 3.2 道路上不同位置的照明强度




```
> Cmin := subs(x = Xm, C);
```

$$Cmin := 18.24392572$$

```
> Xe := fsolve(S[1] = S[2], x, 0..s);
```

$$Xe := 9.003061731$$

```
> dX := Xm - Xe;
```

$$dX := .335237405$$

十分有趣的是，X 点的位置不同于两盏灯照明强度相等的 x_e 点的位置.

对于这个数值例子，我们也能够直接确定点 X. 我们可以把命令 solve 应用于方程 (3.2) 以得到整系数 12 次多项式的代数根. 有若干实数解在两个光源之间.

```
> solve(eq, x);
```

$$\begin{aligned}&\text{RootOf}(56_Z^{12} + 1760_Z^{11} - 411975_Z^{10} + 24315000_Z^{9} - 886167750_Z^{8}\\&\quad + 22194630000_Z^{7} - 388140507750_Z^{6} + 4698666870000_Z^{5} - 37664582848875_Z^{4}\\&\quad + 180676117955000_Z^{3} - 393823660188775_Z^{2} - 31640625000_Z + 316406250000)\end{aligned}$$

```
> select(t-> type(t, numeric) and t > 0 and t < s,
>          [allvalues(")]);
```

$$[.02848997038, 9.338299136, 19.97669581]$$

MAPLE 给出了三个极值，还需要确定哪一个是极小值.

```
> map(t -> if subs(x = t, diff(dC, x)) < 0
>     then max else min fi, ");
```

$$[max, min, max]$$

```
> map(t -> subs(x = t, C), "");
```

$$[81.98104008, 18.24392572, 84.47655488]$$

正如所看到的那样关于 X 得到了同样的结果，即 $x = 9.338299136$. 要注意最大照明强度的点的位置靠近两个光源，但不是在它们的正下方.

3.3 改变 h_2 以极大化照明强度

在这一节我们使用与上一节相同的数值，但把第二个光源的高度作为变量，以极大化点 X 的照明强度. 因此 $C(x, h_2)$ 是两个变量的函数.

作为第一步，我们先求函数 $x(h_2)$, 它满足

$$C\big(x(h_2), h_2\big) = \min_{0 \le x \le s} C(x, h_2).$$

为此我们从 $3[m]$ 到 $9[m]$ 改变 h_2 的值，同时对于每个 h_2 的值反复求解与上一节同样的问题.

```
> h[2] := 'h[2]':
> H2 := array(0..30):  # array for the values of h[2]
> X := array(0..30):  # array for the values of x(h[2])
> for i from 0 to 30 do
>    H2[i] := 3 + 6*i/30:
>    X[i] := fsolve(subs(h[2]=H2[i], dC), x, 0..s):
> od:
> H2 := convert(H2, list):
> X := convert(X, list):
```

图 3.3 是函数 $x(h_2)$ 的图形，它是由如下的命令得到的:

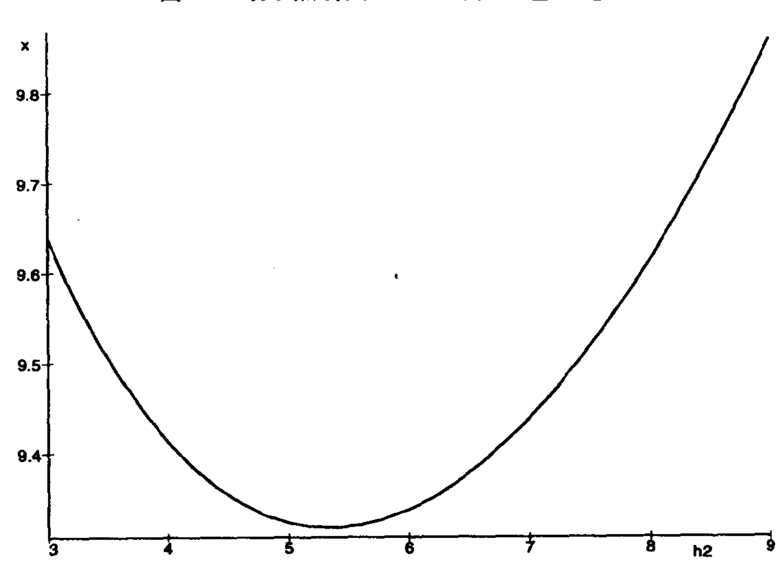
```
> plot(zip((h2, x) -> [h2, x], H2, X), 3..9);
```

我们将把 $C(x, h_2)$ 绘制成三维图，同时把空间曲线 $C\big(x(h_2), h_2\big)$ 也画在图上，这条线上的点都具有最小的照明度. 正如我们所想象的那样，这条空间曲线位于 $C(x, h_2)$ 的谷底.

```
> f := unapply(C, x, h[2]):
> Cu := [seq([X[i], H2[i], f(X[i], H2[i])], i=1..31)]:
> with(plots):
> PL1 := spacecurve(Cu, thickness = 2):
> PL2 := plot3d(subs(h[2] = h2 ,C), x = -s/2..3*s/2, h2 = 3..9,
>              style = wireframe):
> display({PL1, PL2});
```

第二步我们求出曲线 $x(h_2)$ 上最大照明度的点. 这个点在函数 $C(x, h_2)$ 的稳定点当中，也就是说在函数 $C(x, h_2)$ 的梯度等于零的点当中.

图 3.3 最小照明点 x 的坐标， $3 \le h_2 \le 9$.




```
> with(linalg):
> g := grad(C, [x, h[2]]);
```

$$g := \left[-30000\frac{x}{(25+x^2)^{5/2}} - 4500\frac{h_2\,(-40+2\,x)}{({h_2}^2+(20-x)^2)^{5/2}},\right.$$
$$\left.\frac{3000}{({h_2}^2+(20-x)^2)^{3/2}} - 9000\frac{{h_2}^2}{({h_2}^2+(20-x)^2)^{5/2}}\right]$$

```
> Sol := fsolve({g[1] = 0, g[2] = 0}, {x, h[2]},
>               {x = 0..s, h[2] = 3..9});
```

$$Sol := \{x = 9.503151310,\ h_2 = 7.422392890\}$$

为了检验这个解是最大的，我们观察函数 $C(x,h_2)$ 的 Hessian 矩阵在上述点的特征值.

```
> H := subs(Sol, hessian(C, [x, h[2]]));
```

$$H := \begin{bmatrix} 1.056539134 & -.1793442077 \\ -.1793442077 & -.2536310107 \end{bmatrix}$$

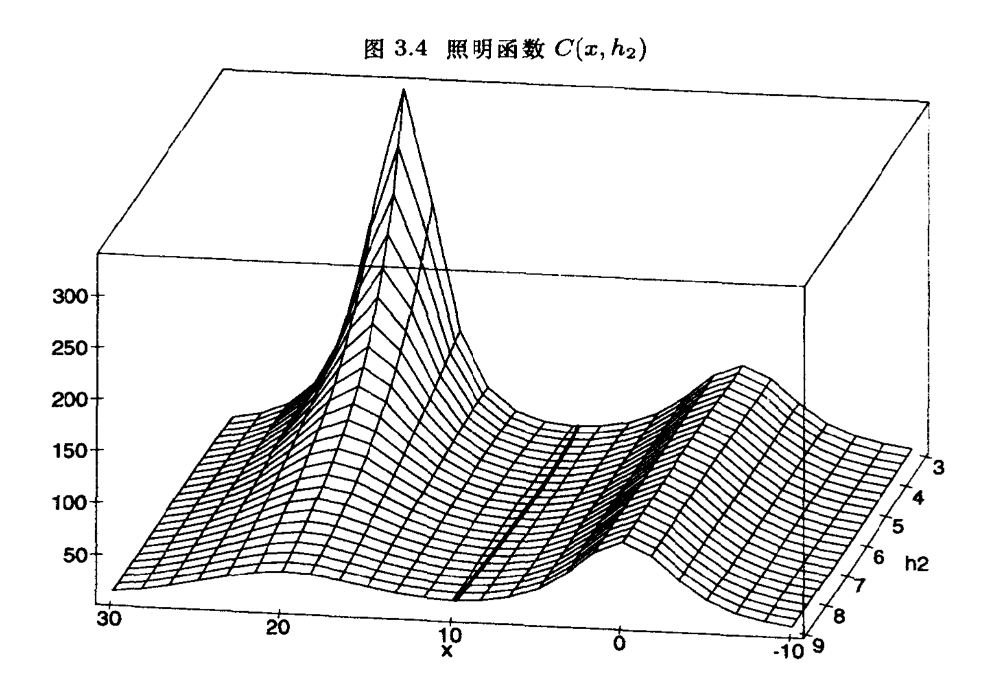
```
> eigenvals(H);
```

$$-.2777372198,\ 1.080645343$$

特征值的不同的符号告诉我们这个点是函数的鞍点. 因为 $H[1,1] = \partial^2 C/\partial x^2 \doteq 1.06 > 0$ 这个点在 x 方向上有极小值，同时因为 $H[2,2] = \partial^2 C/\partial h_2^2 \doteq -0.25 < 0$ 该点在 h_2 方向有极大值，因此我们找到了所求的点，即在所有最小照明点 $(x(h_2),0)$ 当中照明度最大的点.

注意到在这个数值例子中我们能够关于 h_2 解析地求解第二个方程 $g_2 = 0$.

图 3.4 照明函数 $C(x, h_2)$




```
>{solve(g[2] = 0, h[2])};
```

$$\{-\frac{1}{2}\sqrt{2}(-20+x), \frac{1}{2}\sqrt{2}(-20+x)\}$$

这意味着这个解给出了第二个灯的最优高度，投射角度 α_2 由下面计算给出

```
> tan(alpha[2]) = normal("[1]/(s-x));
```

$$\tan(\alpha_2) = \frac{1}{2}\sqrt{2}$$

```
> evalf(arctan(rhs(")));
```

$$.6154797085$$

```
> evalf(convert(", degrees));
```

$$35.26438965 \; degrees$$

或 $\alpha_2 = 35°15'51.8208''$.

3.4 照明优化

道路上的照明均匀是非常重要的．这个问题使用点光源是不能够解决的．我们将总是在光源下方得到最大的照明而在它们之间某处的照明最小．但是对于给定亮度和给定间隔的光源，我们能够通过调整光源的高度来使最小照明强度的点的照明强度达到最大．这一节我们将考虑这个问题．

整个照明情况现在是一个三个变量的函数 $C(x, h_1, h_2)$. 与低维的情况一样，最低照明度的点仍然是由 C 的梯度的根确定. 我们试图求出它的通解；在 MAPLE 中这意味着没有指定我们在数值例子中使用的变量.

```
> P[1] := 'P[1]':  P[2] := 'P[2]':  h[1] := 'h[1]':
> h[2] := 'h[2]':  s := 's':
> g := grad(C, [x, h[1], h[2]]);
```

$$g := \left[-3\frac{P_1 h_1 x}{({h_1}^2+x^2)^{5/2}} - \frac{3}{2}\frac{P_2 h_2(-2s+2x)}{({h_2}^2+(s-x)^2)^{5/2}}, \frac{P_1}{({h_1}^2+x^2)^{3/2}} - 3\frac{P_1 {h_1}^2}{({h_1}^2+x^2)^{5/2}}, \frac{P_2}{({h_2}^2+(s-x)^2)^{3/2}} - 3\frac{P_2 {h_2}^2}{({h_2}^2+(s-x)^2)^{5/2}}\right]$$

MAPLE 不能够解析地求出方程 $g=0$ 的根. 然而我们能够关于 h_1 求解第二个方程，关于 h_2 求解第三个方程.

```
> sh1 := {solve(g[2] = 0, h[1])};
```

$$sh1 := \{-\frac{1}{2}\sqrt{2}\,x, \frac{1}{2}\sqrt{2}\,x\}$$

```
> sh2 := {solve(g[3] = 0, h[2])};
```

$$sh2 := \{-\frac{1}{2}\sqrt{2}\,(s-x), \frac{1}{2}\sqrt{2}\,(s-x)\}$$

我们只对正值感兴趣，因为负值解意味着光源从下面照明道路，这是十分罕见的.

```
> ho[1] := sh1[2];
```

$$ho_1 := \frac{1}{2}\sqrt{2}\,x$$

```
> ho[2] := sh2[2];
```

$$ho_2 := \frac{1}{2}\sqrt{2}\,(s-x)$$

注意到每个灯的最优高度不依赖于光源的亮度. 这个结果定义了一个几何！因此投射角的计算与 3.3 节的计算相同，也就是说

$$\tan\alpha_1 = \tan\alpha_2 = \frac{\sqrt{2}}{2} \Rightarrow \alpha_1 = \alpha_2 = 35°15'51.8028''.$$

把最优高度带入 g_1 我们能够求出所有的解. 我们把实解赋值给变量 Xo ，因为只有它有实际意义.

```
> Q := subs(h[1] = ho[1], h[2] = ho[2], g[1]);
```

$$Q := -\frac{4}{9}\frac{P_1 x^2\sqrt{3}}{(x^2)^{5/2}} - \frac{2}{9}\frac{P_2(s-x)\sqrt{3}(-2s+2x)}{((s-x)^2)^{5/2}}$$

```
> G := simplify(Q,symbolic);
```

$$G := -\frac{4}{9}\frac{(P_1 s^3 - 3P_1 s^2 x + 3P_1 s x^2 - P_1 x^3 - P_2 x^3)\sqrt{3}}{(s-x)^3 x^3}$$

使用如下命令可以直接求解方程 $G = 0$.

```
> Xsols := [solve(G = 0, x)];
```

$$Xsols := [(\%2 - \%1 + \frac{P_1}{P_1 + P_2})\, s,\ (-\frac{1}{2}\,\%2 + \frac{1}{2}\,\%1 + \frac{P_1}{P_1 + P_2} + \frac{1}{2}\, I\,\sqrt{3}\,(\%2 + \%1))\, s,$$
$$(-\frac{1}{2}\,\%2 + \frac{1}{2}\,\%1 + \frac{P_1}{P_1 + P_2} - \frac{1}{2}\, I\,\sqrt{3}\,(\%2 + \%1))\, s]$$
$$\%1 := \frac{P_1\, P_2}{(P_1 + P_2)\,(P_1\, {P_2}^2)^{1/3}}$$
$$\%2 := \frac{(P_1\, {P_2}^2)^{1/3}}{P_1 + P_2}$$

```
> Xsol := remove(has, Xsols, I);
```

$$Xsol := \left[\left(\frac{(P_1\, {P_2}^2)^{1/3}}{P_1 + P_2} - \frac{P_1\, P_2}{(P_1 + P_2)\,(P_1\, {P_2}^2)^{1/3}} + \frac{P_1}{P_1 + P_2}\right) s\right]$$

```
> Xsol := simplify(Xsol, symbolic);
```

$$Xsol := \left[-\frac{(-{P_1}^{2/3}\, {P_2}^{4/3} + P_1\, P_2 - {P_1}^{4/3}\, {P_2}^{2/3})\, s}{(P_1 + P_2)\, {P_1}^{1/3}\, {P_2}^{2/3}}\right]$$

```
> assume(p1>0, p2>0);
> Xo := subs(P[1] = p1^3, P[2] = p2^3, Xsol[1]):
> Xo := simplify(Xo);
```

$$Xo := \frac{s\, p1\tilde{}}{p1\tilde{} + p2\tilde{}}$$

```
> Xo := subs(p1=P[1]^(1/3), p2=P[2]^(1/3), Xo);
```

$$Xo := \frac{s\, {P_1}^{1/3}}{{P_1}^{1/3} + {P_2}^{1/3}}$$

给定 P_1, P_2 和 s, 使用一些简单的推导可以得出优化照明问题的几何结构及其物理解释.

```
> eq1 := op(1,Q)^2 = normal(op(2,Q))^2;
```

$$eq1 := \frac{16}{27}\,\frac{{P_1}^2}{x^6} = \frac{16}{27}\,\frac{{P_2}^2}{(s - x)^6}$$

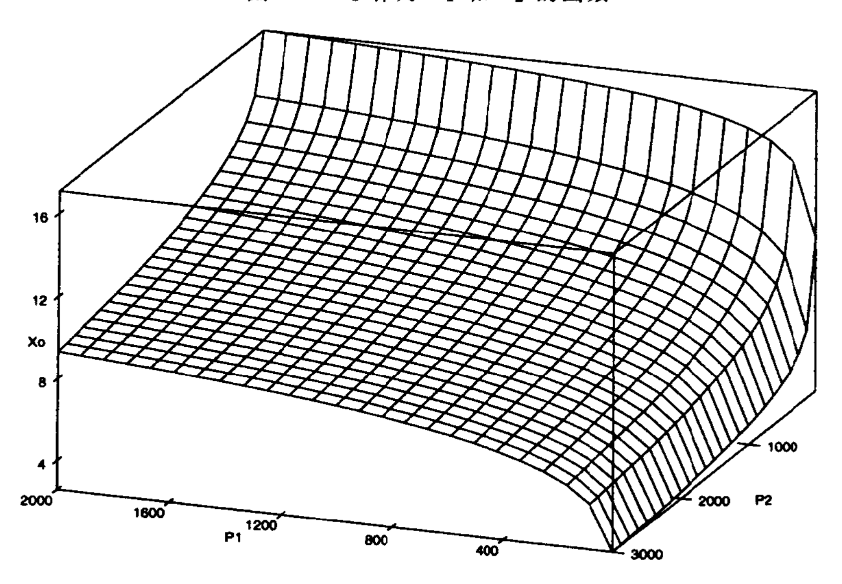
```
> eq2 := simplify(eq1*27*x^6*(s-x)^6/16,symbolic);
```

$$eq2 := (s - x)^6\, {P_1}^2 = x^6\, {P_2}^2$$

```
> eq3 := simplify(map(t -> t^(1/6), eq2), symbolic);
```

$$eq3 := (s - x)\, {P_1}^{1/3} = x\, {P_2}^{1/3}$$

图 3.5 X_0 作为 P_1 和 P_2 的函数



最后一步化简是合理的，因为已知 P_1, P_2, x 是正的且 $x < s$. 所得到的方程有如下的解释：最小照明点的最大照明度将在这样的点得到，只要这个点到两个光源点之间距离 x 和 $s-x$ 的商等于两个光源亮度的立方根之商. 换句话说，如果在某个点上两个光源照射的亮度的体密度相等，则会得到最小照明点中可能最大的照明强度. 方程 eq3 的解也等于 Xo.

在 $s = 20$ 的情形下，我们绘制出最优照明点 Xo 关于两个光源的亮度的函数. 于是我们就可以找出 $P_1 = 2000$ 和 $P_2 = 3000$ 情形下的最优高度.

```
> s := 20:
> plot3d(subs(P[1] = p1, P[2] = p2, Xo), p1 = 0..2000,
>        p2 = 0..3000, orientation = [110, 60], axes=BOXED);
> P[1] := 2000:  P[2] := 3000:  x:= Xo:
> h[1] := ho[1];
```

$$h_1 := 10\,\frac{\sqrt{2}\,2000^{1/3}}{2000^{1/3} + 3000^{1/3}}$$

```
> evalf(");
```

$$6.593948668$$

```
> h[2] := ho[2];
```

$$h_2 := \frac{1}{2}\,\sqrt{2}\,(20 - 20\,\frac{2000^{1/3}}{2000^{1/3} + 3000^{1/3}})$$

```
> evalf(");
```

$$7.548186950$$

正如我们在图 3.5 中看到的，对于很宽的一个亮度值的范围，最优照明点位于 $s/2 = 10$ 附近. 这一点同样可以通过比较 Xo 和第一节计算的最小照明点 Xm 的数值看出. 它们的相对差是

```
> evalf((Xm - Xo)/Xm );
```

$$.0013972025$$

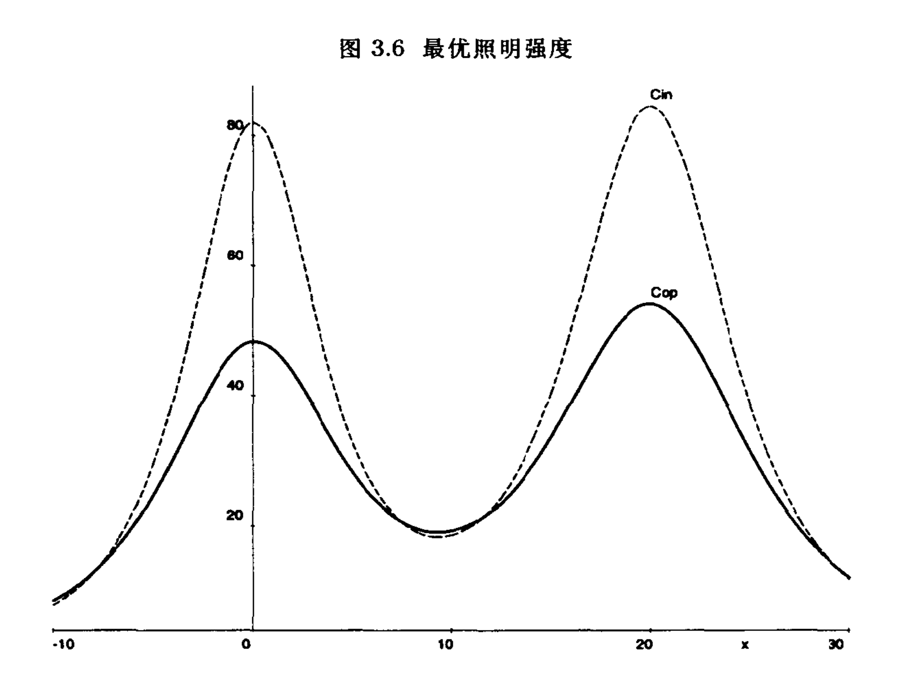
但是照明强度的相对差是

```
> evalf((C - Cmin)/Cmin );
```

$$.040658248$$

要比前者大近 30 倍.

图 3.6 最优照明强度



最后在图 3.6 中我们比较最优解 Cop 和 3.2 节中我们使用固定的高度 $h_1 = 5$ 和 $h_2 = 6$ 所确定的解.

```
> Cop := evalf(C);
```

$$Cop := 18.98569177$$

```
> x :='x':  Cop := C: h[1] := 5:  h[2] := 6:
> plot({C, Cop}, x = -s/2..3*s/2);
```

3.5 结论

基于我们的计算，我们能够对有关照明问题给出如下的建议：

- 如果我们把 Xo 带入最优高度，则它们将仅仅依赖于光源的亮度 P_1, P_2 和它们之间的距离 s.

$$ho_1 = \frac{1}{2}\frac{\sqrt{2}sP_1^{1/3}}{P_1^{1/3}+P_2^{1/3}}, \quad ho_2 = \frac{1}{2}\frac{\sqrt{2}sP_2^{1/3}}{P_1^{1/3}+P_2^{1/3}}.$$

- 对于 $P_1 = P_2$ 的特殊情况，两盏路灯的最优高度是

$$ho_1 = ho_2 = \frac{s}{\sqrt{8}}.$$

参考文献

[1] M.E. GETTYS AND F.J. KELLER, *Classical and modern Physics*, Mc. Graw Hill, 1989.

[2] R.G. LERNER AND G.L. TRIGG, *Encyclopedia of physics*, VCH Publishers, New York, 1991.

第四章　平面三体问题的轨道

D.Gruntz and J.Waldvogel

4.1　引言

平面三体问题是描述平面上三个质点在它们彼此间的牛顿引力作用下的运动的问题．它是常微分方程组数值积分的一个著名应用，因为这个方程组的解用已知的函数来描述十分复杂．

顺便指出，三体问题是一个具有很长历史和大量应用的古典问题 (参见 Szebehely[9] 和 Marchal[4] 的详尽论述)．不过，三体复杂的相互作用经常被人们用二体相互作用的术语来描述，这样从定性的角度看更易于理解．大约一百年前，为了解决这个问题法国科学院曾设立了一笔奖金，这笔奖金后来奖给了 Sundman[7]，他给出了三体问题的收敛级数解．然而由于 Sundman 的级数的收敛速度极慢，对于运动轨道的讨论没有什么实用价值．

这一章，我们将展示如何使用 MAPLE 和 MATLAB 构成和显示三体问题的数值解．在 4.2 节我们将直接使用运动微分方程和 MATLAB 的数值积分器．虽然对于许多初始条件，这个方法很快产生解的初始部分，但对于充分靠近的两个物体，由于对应碰撞的奇异性，这个方法往往是失效的．

在经典的天体力学中， T. Levi-Civita[3] 提出的正则化变换是克服数值积分在两个物体碰撞或接近碰撞时所发生问题的一个有效的技术．因为三个物体能够结成三个物体对， Szebehely 和 Peters[8] 提出当相互距离小于一个确定界限时把 Levi-Civita 变换用于最靠近的一对物体．

在 4.3 节我们将使用由 Waldvogel[10] 提出来的一组变量，它自动地正则化了所出现的三种相遇情况的每一种．由于运动方程变换的复杂性，我们将使用 Hamilton 形式体系得出这些方程．然后运用 MAPLE 的微分算法 (自动微分) 能力生成运动的正则化方程．

4.2　物理坐标下的运动方程

令 $m_j > 0$ $(j = 0, 1, 2)$ 为三个物体的质量， $x_j \in \mathbf{R}^2$ 和 $\dot{x}_j \in \mathbf{R}^2$ 是在惯性坐标系下它们的位置和速度 (列) 向量 (字母上的圆点表示关于时间 t 的导数)．我们将用 r_j 表示两个物体之间的距离 (图 4.1)

$$r_0 = |x_2 - x_1|, r_1 = |x_0 - x_2|, r_2 = |x_1 - x_0|. \tag{4.1}$$

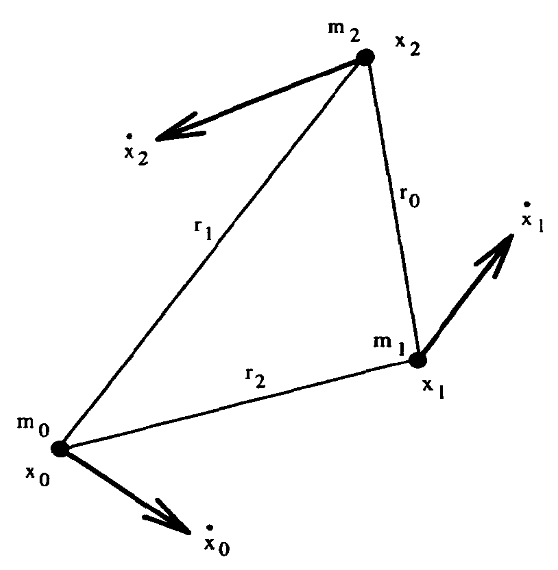
下面我们用 F 表示 m_1 作用在 m_0 上的牛顿引力，则有

$$F = -m_0 m_1 \frac{x_0 - x_1}{|x_0 - x_1|^3}$$

如果选择长度、时间和质量单位，使得引力常数等于 1, 则牛顿运动方程 (最简形式) 变为

$$\begin{aligned}
\ddot{x}_0 &= m_1 \frac{x_1 - x_0}{r_2^3} + m_2 \frac{x_2 - x_0}{r_1^3} \\
\ddot{x}_1 &= m_2 \frac{x_2 - x_1}{r_0^3} + m_0 \frac{x_0 - x_1}{r_2^3} \\
\ddot{x}_2 &= m_0 \frac{x_0 - x_2}{r_1^3} + m_1 \frac{x_1 - x_2}{r_0^3}
\end{aligned} \tag{4.2}$$

图 4.1 在物理坐标下的三体问题



这个有 12 个自由度的方程组可由 MATLAB 以直接的方式积分，定义 (列) 向量 $Y \in \mathbf{R}^{12}$ 为因变量，即

$$Y = [x_0; \dot{x}_0; x_1; \dot{x}_1; x_2; \dot{x}_2].$$

使用算法 4.1 提供的 MATLAB 函数 `Ydot=f(t,Y)`, 可以求解方程组 (4.2).

算法 4.1 函数 Ydot

```
function Ydot=f(t,Y)

global m0 m1 m2 % masses of the three bodies

x0=Y(1:2);x1=Y(5:6);x2=Y(9:10);
d0=(x2-x1)/norm(x2-x1)^3;
d1=(x0-x2)/norm(x0-x2)^3;
d2=(x1-x0)/norm(x1-x0)^3;

Ydot( 1: 2)=Y( 3: 4);
Ydot( 5: 6)=Y( 7: 8);
Ydot( 9:10)=Y(11:12);
Ydot( 3: 4)=m1*d2-m2*d1;
Ydot( 7: 8)=m2*d0-m0*d2;
Ydot(11:12)=m0*d1-m1*d0;
Ydot=Ydot(:);
```

调用 MATLAB 的积分器 `ode113` 和几行 MATLAB 代码就可以产生三个物体的轨道.

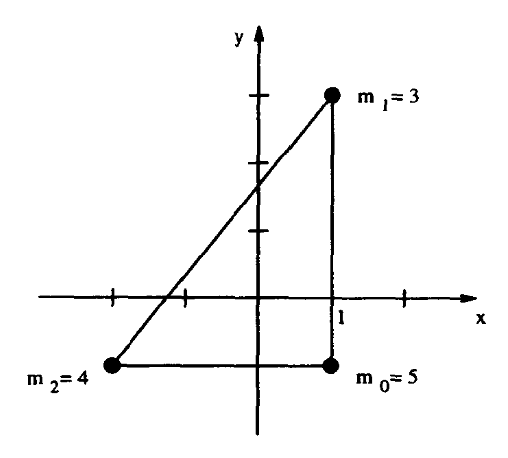
```
>> global m0 m1 m2;
>> m0 = 5; m1 = 3; m2 = 4;
>> x00 = [1;-1]; x10 = [1;3]; x20 = [-2;-1]; xp0 = [0;0];
>> options= odeset('RelTol',1e-10,'AbsTol',1e-10);
>> [T1,Y1] = ode113('f', [0 63], [x00;xp0;x10;xp0;x20;xp0],
                    options);
```

在上面的例子中，由于历史原因使用了所谓的 Pythagorean 初始条件 (参见图 4.2),

$$\begin{array}{rclrclrcl} m_0 & = & 5, & m_1 & = & 3, & m_2 & = & 4, \\ x_0 & = & \begin{pmatrix} 1 \\ -1 \end{pmatrix}, & x_1 & = & \begin{pmatrix} 1 \\ 3 \end{pmatrix}, & x_2 & = & \begin{pmatrix} -2 \\ -1 \end{pmatrix}. \\ \dot{x}_0 & = & 0, & \dot{x}_1 & = & 0, & \dot{x}_2 & = & 0 \end{array} \tag{4.3}$$

这些数据于 1913 年首先被 Burrau 使用 [1], 但是这个系统的最后发展是于 1967 年由 Szebehely 和 Peters [8] 使用精细的数值积分解决的. 这个问题的历史记载同样在 [4] 中给出, 这些并没有直接的天文或物理的意义.


图 4.2 Pythagorean 问题的初始构形



对于时间区间 $0 \le t \le 63$, 在配置为 200 MHz Intel Pentium Pro 处理器的 PC 上运行了大约三分钟的时间执行了 11016 次积分运算. 在图 4.3 和 4.4 中我们给出了三个物体在时间区间 $0 \le t \le 10$ 和 $10 \le t \le 20$ 的轨道, 同时最后在图 4.5 中给出了区间 $50 \le t \le 63$ 上的轨道.

```
>> R1 = 1:1900;
>> plot(Y1(R1,1),Y1(R1,2),'-',...
        Y1(R1,5),Y1(R1,6),':', Y1(R1,9),Y1(R1,10),'-.')
>> R2 = 1901:3633;
>> plot(Y1(R2,1), Y1(R2,2),'-',...
        Y1(R2,5),Y1(R2,6),':', Y1(R2,9),Y1(R2,10),'-.')
>> R3 = 8515:11016;
>> plot(Y1(R3,1),Y1(R3,2),'-',...
```

```
        Y1(R3,5),Y1(R3,6),':', Y1(R3,9),Y1(R3,10),'-.')
```

在上面的例子中使用的最小步长是

```
>> [m,k] = min(diff(T1));
>> m


m =
  8.8727e-008
>> T1(k)
ans =
  15.8299
```

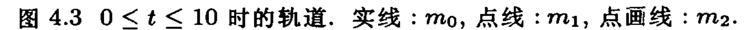
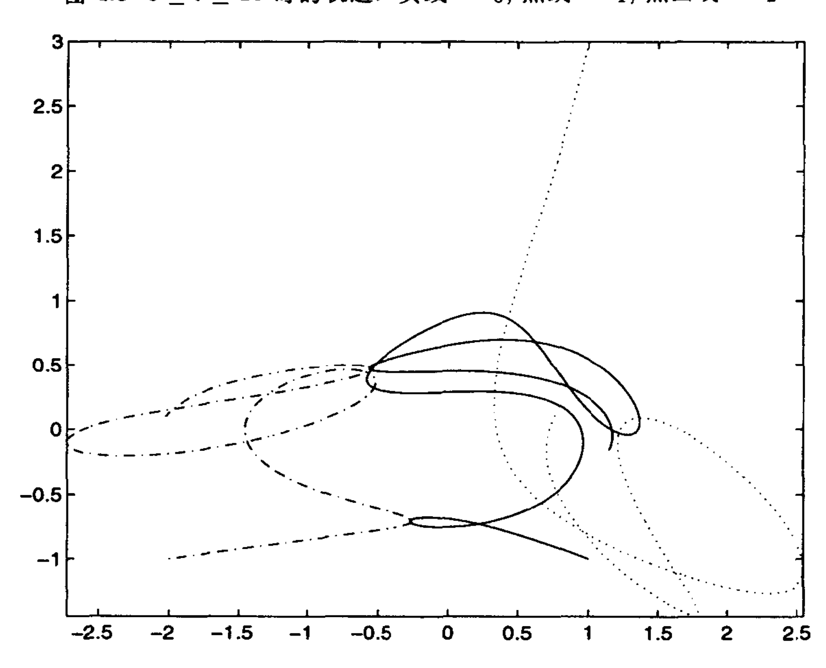
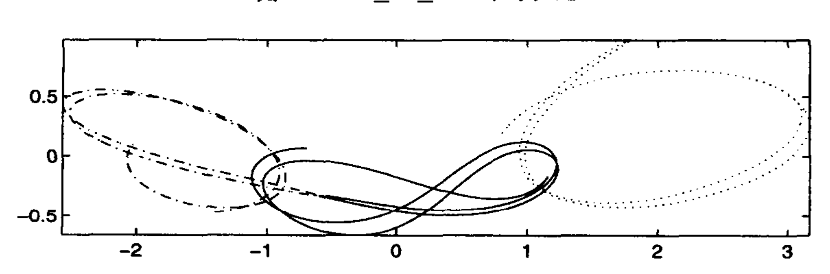


图 4.3 $0 \le t \le 10$ 时的轨道. 实线：m_0, 点线：m_1, 点画线：m_2.



这个最小的步长是在 $t = 15.8299$ 时需要的，这时 m_0 和 m_2 之间出现了接近碰撞的现象. 积分器

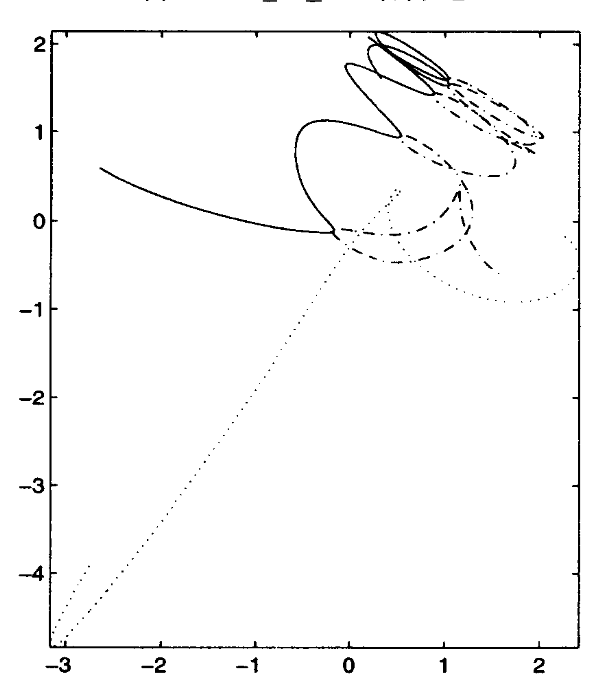
图 4.4 $10 \le t \le 20$ 时的轨道.



几乎不能处理这种接近碰撞的情况. 然而对于 $t > 15.83$, 尽管容许误差小到 $1e-10$, 轨道的精确性也是相当低的, 这一点可以从把图 4.5 与图 4.9 和 [8] 中的结论相比较看出.

事实上由这个数值积分预报的最终演化结果是不正确的. 为了补偿接近碰撞时的高速度, 这个积分器必须急剧地减小积分的步长, 有时甚至要接近导致不精确结果的容许精确度的水平. 这些问题将在下一节通过引入新的变量来解决, 这些变量将使所有可能的二元碰撞出现的奇异性正则化.

图 4.5 $50 \le t \le 63$ 时的轨道



4.3 全局正则化

为了简单起见, 我们将假设质心于初始时在原点处于静止状态, 即

$$\sum_{j=0}^{2} m_j x_j = 0, \qquad \sum_{j=0}^{2} m_j \dot{x}_j = 0; \tag{4.4}$$

则运动方程 (4.2) 意味着在整个运动过程中 (4.4) 是满足的. 这个事实可以删除其中一个变量 x_j. 这可以在 Hamilton 形式体系下通过引入关于 x_0 的相对坐标

$$X = x_1 - x_0, \qquad Y = x_2 - x_0 \tag{4.5}$$

来实现, 就象典型共轭矩那样

$$P = m_1 \dot{x}_1, \qquad Q = m_2 \dot{x}_2. \tag{4.6}$$

在相对坐标下这些质点之间的距离就变为

$$r_0 = |Y - X|, \qquad r_1 = |Y|, \qquad r_2 = |X|. \tag{4.7}$$

于是运动方程可以从 Hamilton 函数

$$H=\frac{|P+Q|^2}{2m_0}+\frac{|P|^2}{2m_1}+\frac{|Q|^2}{2m_2}-\frac{m_1m_2}{|Y-X|}-\frac{m_0m_1}{|X|}-\frac{m_0m_2}{|Y|} \tag{4.8}$$

得到，为

$$\dot{X}=\frac{\partial H}{\partial P},\qquad \dot{Y}=\frac{\partial H}{\partial Q},\qquad \dot{P}=-\frac{\partial H}{\partial X},\qquad \dot{Q}=\frac{\partial H}{\partial Y}. \tag{4.9}$$

三个物体的坐标 x_0,x_1,x_2 可以由方程 (4.4) 和 (4.5) 被恢复为

$$x_0=-\frac{m_1X+m_2Y}{m_0+m_1+m_2},\quad x_1=x_0+X,\quad x_2=x_0+Y. \tag{4.10}$$

下面为方便起见，与向量 $v=(v_1,v_2)^{\mathrm{T}}\in\mathbf{R}^2$ 有关联的复数 v_1+iv_2 将被用相同的符号 $v\in\mathbf{C}$ 来表示．在两种记法下惯用的模的符号 $|v|=\sqrt{v_1^2+v_2^2}$ 也在这里使用．

为了正则化在 m_0 和 m_1 碰撞时的运动方程， Levi-Civita 方法需要按照保角映射 $X=x^2$ 引入新的复坐标 $x\in\mathbf{C}$ 以代替 $X\in\mathbf{C}$. 进而，必须按照微分之间的关系 $\mathrm{d}t=|X|\mathrm{d}s$ 引入一个新的虚构时间 s [3].

在三体问题中为了同时正则化所有三个碰撞，我们将使用由 Waldvogel 提出的这个变换的推广 [10]. 为了代替复坐标 X,Y 和时间 t, 我们使用新的坐标 $x\in\mathbf{C},y\in\mathbf{C}$ 和虚构时间 s, 它们与 X,Y,t 的关系是

$$X=\left(\frac{x^2-y^2}{2}\right)^2,\qquad Y=\left(\frac{x^2+y^2}{2}\right),\qquad \mathrm{d}t=r_0r_1r_2\mathrm{d}s. \tag{4.11}$$

变换 (4.11) 的正则化效果的关键是关系式

$$Y-X=(xy)^2; \tag{4.12}$$

这样所有三个复相对坐标 $X,Y,X-Y$ 被写成新的复坐标的完全平方形式.

把运动方程 (4.9) 变换为新变量的机制需要引入新的矩量 $p\in\mathbf{C},q\in\mathbf{C}$, 使得从变量组 (X,Y,P,Q) 变换的变量 (x,y,p,q) 是典型的．定义新矩量关系的结果是 [10]

$$\begin{pmatrix}p\\q\end{pmatrix}=\bar{A}^{\mathrm{T}}\begin{pmatrix}P\\Q\end{pmatrix} \tag{4.13}$$

其中

$$A=\begin{pmatrix}\frac{\partial X}{\partial x}&\frac{\partial X}{\partial y}\\\frac{\partial Y}{\partial x}&\frac{\partial Y}{\partial y}\end{pmatrix}=\begin{pmatrix}x(x^2-y^2)&-y(x^2-y^2)\\x(x^2+y^2)&y(x^2+y^2)\end{pmatrix} \tag{4.14}$$

$\bar{A}^{\mathrm{T}}$ 表示 A 的复共轭转置矩阵.

正则化的运动方程是

$$\frac{\mathrm{d}x}{\mathrm{d}s}=\frac{\partial K}{\partial p},\quad \frac{\mathrm{d}y}{\mathrm{d}s}=\frac{\partial K}{\partial q},\quad \frac{\mathrm{d}p}{\mathrm{d}s}=-\frac{\partial K}{\partial x},\quad \frac{\mathrm{d}q}{\mathrm{d}s}=-\frac{\partial K}{\partial y},\quad \frac{\mathrm{d}t}{\mathrm{d}s}=r_0\,r_1\,r_2, \tag{4.15}$$

其中

$$K=r_0\,r_1\,r_2\,(H-E) \tag{4.16}$$

是以新变量表示的正则化的 Hamilton 函数，同时 E 是 H 的初始值，即总的能量．因为在轨道上 $H(t)=E=$ 常量，可得在这个轨道上有 $K(s)=0$.

在进一步的操作中，将尽量多地使用 MAPLE 的功能．然而为求 K 的梯度使用符号微分将使 (4.15) 的右端产生许多无效赋值的项．

下面介绍一个成功的方法．第一步使用自然出现的辅助量，如 r_0, r_1, r_2 等，将 K 的表达式写得尽量简捷．这时 K 的梯度就可以借助于自动微分法赋值从而产生有效的编码．

令 $x = x_1 + \mathrm{i}x_2$, $y = y_1 + \mathrm{i}y_2$, $x_1 \in \mathbf{R}, x_2 \in \mathbf{R}$. 则按照 (4.11) 和 (4.7) 我们得到

```
> x := x1 + I*x2:
> y := y1 + I*y2:
> X := ((x^2-y^2)/2)^2:
> Y := ((x^2+y^2)/2)^2:
> r0 := factor(evalc(abs(Y-X))):
> r0 := simplify(r0, power, symbolic);
```

$$r0 := (x2^2 + x1^2)(y2^2 + y1^2)$$

```
> r1 := factor(evalc(abs(Y)));
```

$$r1 := \frac{1}{4}(y1^2 - 2\,x2\,y1 + x2^2 + y2^2 + 2\,y2\,x1 + x1^2)(y1^2 + y2^2 + 2\,x2\,y1 + x2^2 - 2\,y2\,x1 + x1^2)$$

```
> r2 := factor(evalc(abs(X)));
```

$$r2 := \frac{1}{4}(y1^2 + y2^2 - 2\,y2\,x2 + x2^2 - 2\,y1\,x1 + x1^2)(y1^2 + y2^2 + 2\,y2\,x2 + x2^2 + 2\,y1\,x1 + x1^2)$$

容易看出因子 r_1 和 r_2 是平方和．我们写了一个小的 MAPLE 过程 reduce (参见第 61 页的算法 4.4) 去化简这些表达式，因为 MAPLE 没有直接的命令实现这个操作．

```
> r1 := reduce(r1);
```

$$r1 := \frac{1}{4}((y1 - x2)^2 + (y2 + x1)^2)((y1 + x2)^2 + (y2 - x1)^2)$$

```
> r2 := reduce(r2);
```

$$r2 := \frac{1}{4}((y1 - x1)^2 + (y2 - x2)^2)((y1 + x1)^2 + (y2 + x2)^2)$$

根据 (4.16)(4.8) 和 (4.7) 我们得到 K 的表达式

$$K = \frac{L_0}{2m_0} + \frac{L_1}{2m_1} + \frac{L_2}{2m_2} - (m_0 m_1 r_0 r_1 + m_1 m_2 r_1 r_2 + m_2 m_0 r_2 r_0) - E r_0 r_1 r_2. \tag{4.17}$$

对于辅助变量 L_j, 经过简单地计算可得

$$\begin{aligned} L_0 &= r_0 r_1 r_2 |P + Q|^2 = \frac{r_0}{16}|\bar{x}p - \bar{y}q|^2 \\ L_1 &= r_0 r_1 r_2 |P|^2 = \frac{r_1}{16}|\bar{y}p - \bar{x}q|^2 \\ L_3 &= r_0 r_1 r_2 |Q|^2 = \frac{r_2}{16}|\bar{y}p + \bar{x}q|^2 \end{aligned} \tag{4.18}$$

使用实数记法 $p=p_1+\mathrm{i}p_2, q=q_1+\mathrm{i}q_2$, 将上述表达式写成以因子形式为主

$$\begin{aligned} L_0 &= \frac{r_0}{16}\left(|p|^2|x|^2+|q|^2|y|^2-A-B\right) \\ L_1 &= \frac{r_1}{16}\left(|p|^2|y|^2+|q|^2|x|^2-A+B\right) \\ L_2 &= \frac{r_2}{16}\left(|p|^2|y|^2+|q|^2|x|^2+A-B\right) \end{aligned}$$

其中

$$\begin{aligned} A &= 2(x_1y_1+x_2y_2)(p_1q_1+p_2q_2) \\ B &= 2(x_2y_1-x_1y_2)(p_2q_1-p_1q_2). \end{aligned}$$

在算法 4.2 中给出了计算正则化 Hamilton 函数 K 的一个 MAPLE 程序，其中 L_j 的意义稍微有所修正，同时总能量 E 用 EE 来表示.

为了计算 K 关于所有变量的偏导数，我们使用 MAPLE 的自动微分功能. 关于自动 (或算法的) 微分法的介绍我们参考了 [2,5]. 我们知道，所谓的自动微分法的反向模式最适宜于计算梯度. 将前向模式应用于 K 将导致大约 300 次乘法的过程 (优化以后), 而反向模式则仅仅约为 200 次.

算法 4.2 计算 K 的程序

```
K:=proc(x1,x2,y1,y2,p1,p2,q1,q2)
  local xx,yy,pp,qq,r0,r1,r2,
        A,B,L0,L1,L2,m01,m12,m20,apb,amb;

  xx:=x1^2+x2^2; yy=y1^2+y2^2;
  pp:=p1^2+p2^2; qq=q1^2+q2^2;

  r0:=xx*yy;
  r1:=((x1-y2)^2+(x2+y1)^2)*((x1+y2)^2+(x2-y1)^2)/4;
  r2:=((x1+y1)^2+(x2+y2)^2)*((x1-y1)^2+(x2-y2)^2)/4;

  A:=2*(p1*q1+p2*q2)*(x1*y1+x2*y2);
  B:=2*(p2*q1-p1*q2)*(x2*y1-x1*y2);

  apb:=A+B;
  amb:=A-B;
  L0:=r0*(pp*xx+qq*yy-apb);
  L1:=r1*(pp*yy+qq*xx-amb);
  L2:=r2*(pp*yy+qq*xx+amb);

  m01:=m0*m1;
  m12:=m1*m2;
  m20:=m2*m0;

  L0/32/m0+L1/32/m1+L2/32/m2-m01*r0*r1-m12*r1*r2-m20*r2*r0-EE*r0*r1*r2;
```

```
end:
```

如果我们在计算导数时事先分开各个乘积以避免产生共同的子表达式，这个计算次数还可以减少.

```
> SK := SPLIT(K):
> RSK := REVERSEMODE(SK):
> ORSK := readlib(optimize)(RSK):
> COST(ORSK);
```

$$138\ multiplications + 129\ assignments + 81\ subscripts + 86\ additions + 3\ divisions + functions$$

我们这里使用的程序可以通过匿名 FTP 从 `ftp.inf.ethz.ch`[1] 得到. 最后的结果是计算方程 (4.15) 右端的 MAPLE 程序 ORSK, 它仅仅带有 144 乘法 (乘法和除法) 运算. [10] 的作者之一写了一个手工优化的程序，它大概需要 100 次乘法运算和 50 次加法运算，因此由 MAPLE 得到的结果几乎是最优的.

用 MAPLE 同样能够证明程序 ORSK 实际上是正确的. 为此我们使用软件包 Linalg 中的 gradient 函数计算梯度，并与程序 ORSK 产生的结果按照符号输入参数逐个元素进行比较.

```
> G1 := linalg[grad](K(x1,x2,y1,y2,p1,p2,q1,q2),
>                     [x1,x2,y1,y2,p1,p2,q1,q2]):
> G2 := [ORSK(x1,x2,y1,y2,p1,p2,q1,q2)]:
> zip((g1,g2)->expand(g1-g2), convert(G1, list), G2);
```

$$[0, 0, 0, 0, 0, 0, 0, 0]$$

为方便起见数值积分和轨道的图形输出将再一次由 MATLAB 实现. 为了将 MAPLE 的 ORSK 程序转换为 MATLAB 程序，我们需要作一些语法上的变化，这可以借助于简单的编辑器来实现. 首先，MAPLE 的赋值 ":=" 必须转换为 MATLAB 的 "=", 同时向量的记法必须由方括号 (a[1]) 转换为圆括号 (a(1)). 还要加上程序头，结果必须以数组的形式存起来. 这就得到了一个大约 150 行的 MATLAB 程序，它被列在算法 4.3 中.

算法 4.3 函数 threebp

```
function yprime=threebp(s,y)

   global m0 m1 m2 EE

   x1=y(1); x2=y(2); y1=y(3); y2=y(4);
   p1=y(5); p2=y(6); q1=y(7); q2=y(8);

   % here comes the Maple generated code
   t1=x1^2;
   ...
   ...
   t137=t43*xx+t44*xx+t45*yy;
```

[1]URL: ftp://ftp.inf.ethz.ch/pub/SolvingProblems/ed3/chap04/

```
    grd(1)=-y2*t96+y1*t98+t100+t101+t102+t103+2*x1*t104;
    grd(2)=y1*t96+y2*t98+t109+t110+t111+t112+2*x2*t104;
    grd(3)=x2*t96+x1*t98-t100+t101-t111+t112+2*y1*t117;
    grd(4)=-x1*t96+x2*t98-t109+t110+t102-t103+2*y2*t117;
    grd(5)=-q2*t124+q1*t126+2*p1*t128;
    grd(6)=q1*t124+q2*t126+2*p2*t128;
    grd(7)=p2*t124+p1*t126+2*q1*t137;
    grd(8)=-p1*t124+p2*t126+2*q2*t137;

    yprime(1:4)=grd(5:8);
    yprime(5:8)=-grd(1:4);
    yprime(9)=r0*r1*r2;
  end
```

因为在正则化变量中时间 t 是因变量，轨道本身是从方程组 (4.15) 的前 8 个微分方程得到的，而三体系统的时间变化可以从最后一个方程得到.

在积分这个微分方程组时，时间判定的工作是关于导数的计算，即上面的程序每步至少执行一次赋值. 自从 MATLAB 程序出现以来，一个好的想法是用 C 语言程序 (要有一些语法上的变化) 来执行这个计算并把它与 MATLAB 动态相连. 这种方式下加速因子可达到 10. 对于 C 代码和 MATLAB 连接更详细的内容可以参考 123 页的 9.3.4 节.

4.4 Pythagorean 三体问题

为了在正则变量下积分出一个轨道，x, y, p, q 在 $s = t = 0$ 的初始值必须使用变换 (4.11) 的反变换

$$x = \sqrt{\sqrt{Y} + \sqrt{X}}, \qquad y = \sqrt{\sqrt{Y} - \sqrt{X}},$$

和 (4.13) 来计算. 对于 Pythagorean 初始条件 (参见方程 (4.3) 和图 4.2) 我们可以再次使用 MAPLE 来求得这些初始值. evalc 函数把一个复数表达式分裂为它的实数和虚数部分，同时函数 radnormal 简化了开方的运算.

```
> Digits := 20:  readlib(radnormal):
> X := 4*I:
> Y := -3:
> x := sqrt(sqrt(Y)+sqrt(X));
```

$$x := \sqrt{I\sqrt{3} + \sqrt{2} + I\sqrt{2}}$$

```
> x := map(t->map(radnormal,t),evalc(x));
```

$$x := \frac{1}{2}\sqrt{2 + 2\sqrt{3}\sqrt{2} + 2\sqrt{2}} + \frac{1}{2}I\sqrt{2 + 2\sqrt{3}\sqrt{2} - 2\sqrt{2}}$$

```
> evalf(");
```

$$1.5594395315555318362 + 1.0087804965427521214\, I$$

```
> y := sqrt(sqrt(Y)-sqrt(X));
```

$$y := \sqrt{I\sqrt{3} - \sqrt{2} - I\sqrt{2}}$$

```
> y := map(t->map(radnormal,t),evalc(y));
```

$$y := \frac{1}{2}\sqrt{-2 + 2\sqrt{3}\sqrt{2} - 2\sqrt{2}} + \frac{1}{2} I \sqrt{-2 + 2\sqrt{3}\sqrt{2} + 2\sqrt{2}}$$

```
> evalf(");
```

$$.13280847188730666477 + 1.1966000386838271257\, I$$

因为物体在初始时是静止的, $p = q = 0$. 总能量 E 可以由条件 $K(0) = 0$ 得到.

```
> m0 := 5:  m1 := 3:  m2 := 4:
> radnormal(solve(K(Re(x), Im(x), Re(y), Im(y), 0, 0, 0, 0), EE));
```

$$\frac{-769}{60}$$

我们现在启动 MATLAB 使用 ode45 积分微分方程组.

```
>> global m0 m1 m2 EE;
>> m0 = 5; m1 = 3; m2 = 4;
>> p10 = 0; q10 = 0; p20 = 0; q20 = 0;
>> x10 = 1.5594395315555318362;  x20 = 1.0087804965427521214;
>> y10 = 0.13280847188730666477; y20 = 1.1966000386838271256;
>> EE = -769/60;
>> options= odeset('RelTol',1e-10,'AbsTol',1e-10);
>> [S,Z] = ode113('threebp', [0 7.98], ...
                     [x10,x20,y10,y20,p10,p20,q10,q20,0], options);
```

对于上面的积分在精确度 10^{-10} 下需要 2768 步积分 (在 Pentium Pro 200 处理器上大约需要一分钟). 虚构时间的区间 $0 \le s \le 7.98$ 相当于 $0 \le t \le t_f = 63.6127$.

```
>> size(Z)

ans =
      2768   9
>> Z(size(Z))

ans =
   63.6127
```

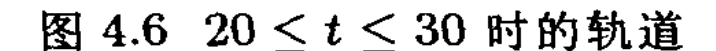
下面我们借助方程 (4.11) 和 (4.10) 恢复三个物体的坐标 x_0, x_1, x_2

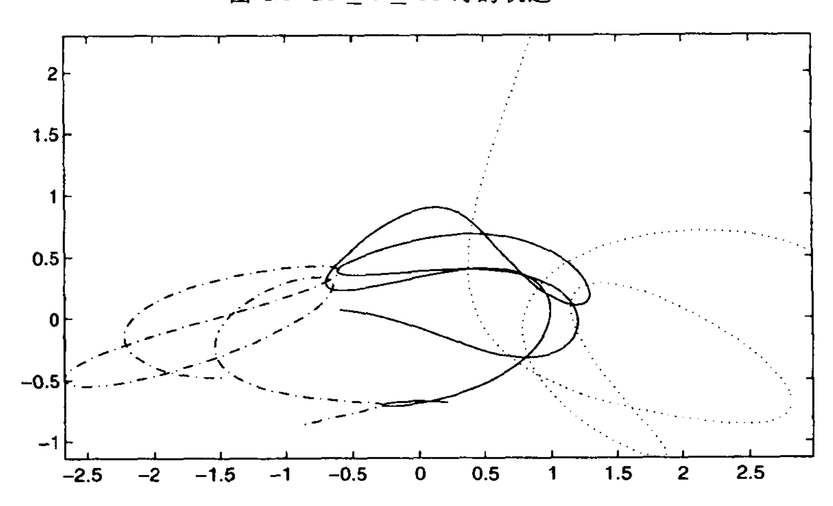
```
>> x = Z(:,1)+i*Z(:,2);
>> y = Z(:,3)+i*Z(:,4);
>> X = (x.^2-y.^2).^2/4;
```

```
>> Y = (x.^2+y.^2).^2/4;
>> x0 = - (m1*X+m2*Y) / (m0+m1+m2);
>> x1 = x0 + X;
>> x2 = x0 + Y;
```

现在我们可以使用如下的命令以动画的形式来观看三个物体的轨道.

```
>> clf
>> axis([-3.3361 3.4907 -4.9904 8.5989])
>> bold on
>> [n,e]=size(x1);
>> for k=1:n-1,
>>    plot(x0(k:k+1),'r-','EraseMode','none');
>>    plot(x1(k:k+1),'g:','EraseMode','none');
>>    plot(x2(k:k+1),'b-.','EraseMode','none');
>>    drawnow
>> end
>> hold off
```


图 4.6 $20 \leq t \leq 30$ 时的轨道



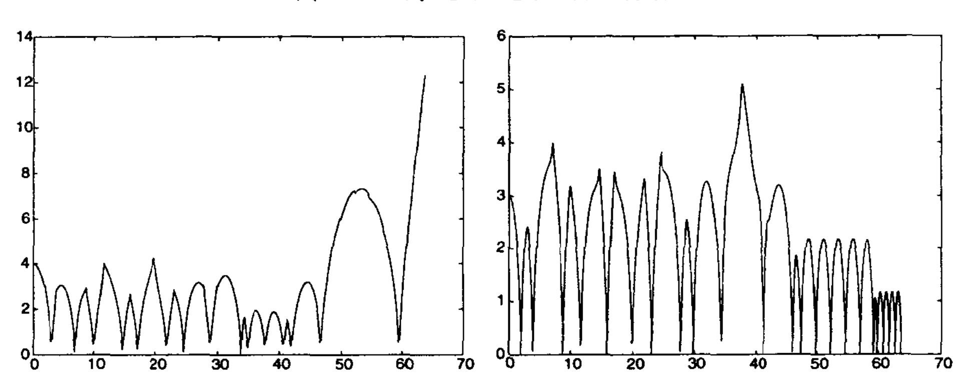
这个动画的几个片段将在下面给出. $0 \leq t \leq 20$ 的轨道与图 4.3 和图 4.4 所示的相同. 其余时间区间的轨道在图 4.6, 图 4.8 和图 4.9 中给出. 在这些图中点线表示 m_1 的轨道, 点划线表示 m_2 的轨道, 实线描述了具有最大质量的 m_0 的运动.

我们已经得到了全部轨道的资料, 可以对这些轨道作进一步地讨论. 例如我们可以求出两个物体接近碰撞的次数. 这可以通过对 r_1 和 r_2 即 X 和 Y 的绝对值随时间 t 变化图形的观察看出.

```
>> T2=Z(:,9);
>> plot(T2, abs(X))
>> plot(T2, abs(Y))
```

在三体运动的系统中, 在 $t \geq 0$ 的任何时间内, 两体之间的最小距离是于 $t = 15.8299$ 在 m_0 和

图 4.7 距离 r_1 和 r_2 随时间的变化



m_2 之间发生的: $r_1 = 9.1094e-04$. 这个接近的碰撞可以在图 4.4 看出, 更详细的情况在图 4.10 中给出.

```
>> [m,k]=min(abs(Y))
m =
  4.4744e-004
k =
  655

>> T2(k)
ans =
    15.8299
```

讨论三体速度向量的一种简单方法是使用两个相邻积分步骤之间的前向差分方程作为近似, 即

$\dot{x}_0$=diff(x0) ./ diff(T).

但使用方程 (4.13) 和 (4.6) 同样有可能从轨道资料得到真实的速度向量. 首先我们借助于 MAPLE 给出 P 和 Q 的符号表达式 (抑制对力矩的复共轭):

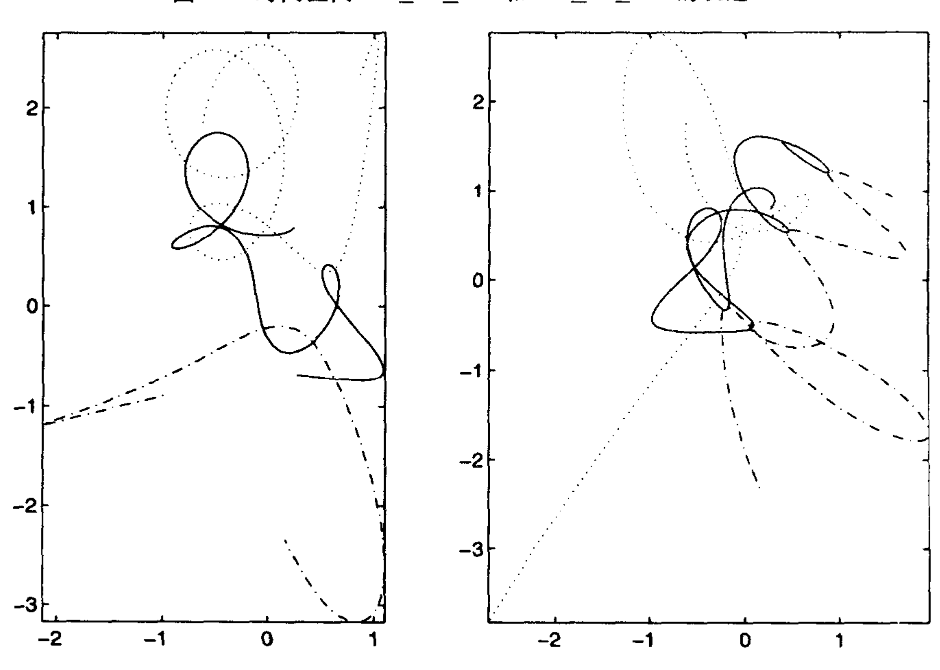
```
> x := 'x':  y := 'y':
> B := linalg[matrix](2,2,[[x*(x^2-y^2), -y*(x^2-y^2)],
>                          [x*(x^2+y^2), y*(x^2+y^2)]]);
```

$$B := \begin{bmatrix} x\,(x^2-y^2) & -y\,(x^2-y^2) \\ x\,(x^2+y^2) & y\,(x^2+y^2) \end{bmatrix}$$

```
> invB := linalg[inverse](transpose(B)):
> pq := linalg[vector]([p,q]):
> PQ := map(normal,linalg[multiply](invB,pq));
```

$$PQ := \left[-\frac{1}{2}\,\frac{-p\,y+q\,x}{x\,(x^2-y^2)\,y},\ \frac{1}{2}\,\frac{p\,y+q\,x}{x\,(x^2+y^2)\,y}\right]$$

图 4.8 时间区间 $30 \le t \le 40$ 和 $40 \le t \le 50$ 的轨道



除了表示共轭的横线外，这个结果与 [10] 中给出的结果是一致的，即

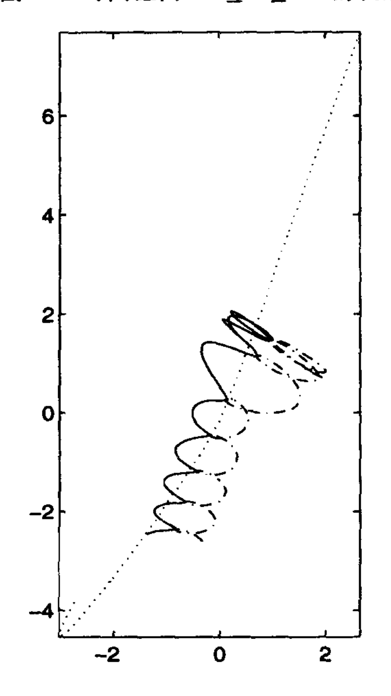
$$P = \frac{p\bar{y} - q\bar{x}}{2\bar{x}\bar{y}(\bar{x}^2 - \bar{y}^2)}, \qquad Q = \frac{p\bar{y} + q\bar{x}}{2\bar{x}\bar{y}(\bar{x}^2 + \bar{y}^2)}. \tag{4.19}$$

我们还能找到最大的速度．结果是在 $t = 15.8299$ 时，m_0 和 m_2 有它们的最大的速度，也就是在 4.2 节我们求出的接近碰撞的位置．在这个时间，m_1 的速度非常小．要注意到这些最大速度是由数值积分产生的离散点集上的最大值．因此如果改变积分器，这些最大值会有相应的变化．

```
>> xbar = Z(:,1)-i*Z(:.2);
>> ybar = Z(:,3)-i*Z(:,4);
>> p = Z(:,5)+i*Z(:,6);
>> q = Z(:,7)+i*Z(:,8);
>> P = (p ./ xbar - q ./ ybar) ./ (2*(xbar.^2 - ybar.^2));
>> Q = (p ./ xbar + q ./ ybar) ./ (2*(xbar.^2 + ybar.^2));
>> vx1 = P/m1;
>> vx2 = Q/m2;
>> vx0 = - (m1*vx1+m2*vx2)/m0;
>> [v,k]=max(abs(vx0))

v =
  89.1372
k =
  655
```

图 4.9 时间区间 $50 \le t \le 63$ 的轨道




```
>> [v,k]=max(abs(vx2))
v =
  111.4273
k =
  655

>> abs(vx1(k))
ans =
    0.0404
```

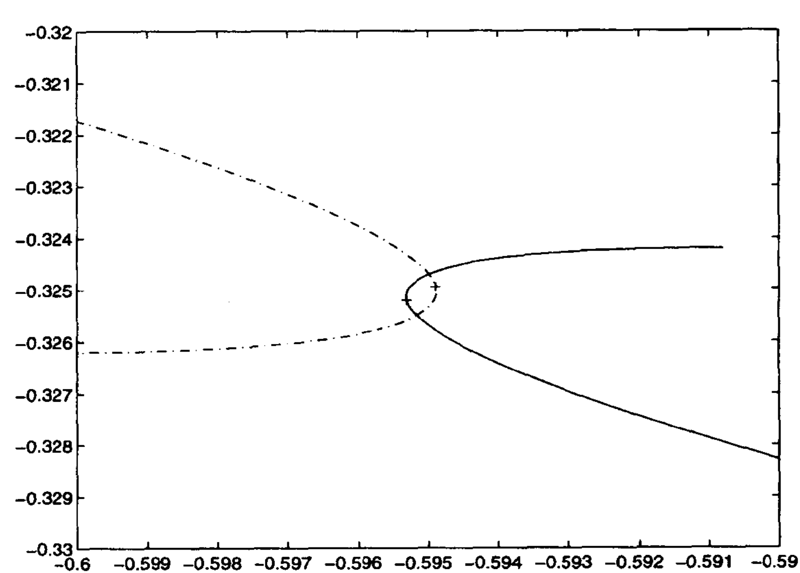
在 $t = 15.8299$ 发生的接近碰撞有着特殊的意义. 注意到, 这三个物体在这次接近相遇的前后近似描述了相同的轨道 (参见图 4.4). 在图 4.6 中仍有这种趋势, 同时在 $t \approx 31.66$ 三个物体以很小的速度又近似占据了它们的初始位置 (参见图 4.8 的左面一幅). 这意味着接近了 Pythagorean 的初始条件, 从而存在着周期解. 在图 4.10 中, 我们看到了图 4.4 的细节, 它是使用如下的命令从 4.2 节得到的数据中产生的:

```
>> [m,k] = min(diff(T1));
>> R = k-200:k+200;
>> plot(Y1(R,1),Y1(R,2),'-',...
        Y1(R,5),Y1(R,6),':',Y1(R,9),Y1(R,10),'-.')
>> hold on
>> plot(Y1(k,1), Y1(k,2), '+')
```

```
>> plot(Y1(k,9), Y1(k,10), '+')
>> axis([-0.60 -0.59 -0.33 -0.32])
>> hold off
```

最后，图 4.9 显示了所有三个物体在接近的相遇之后三条轨道的渐进行为. 要指出，在 4.2 节 (图 4.5) 以直接的方式得到的最终结果是不正确的.

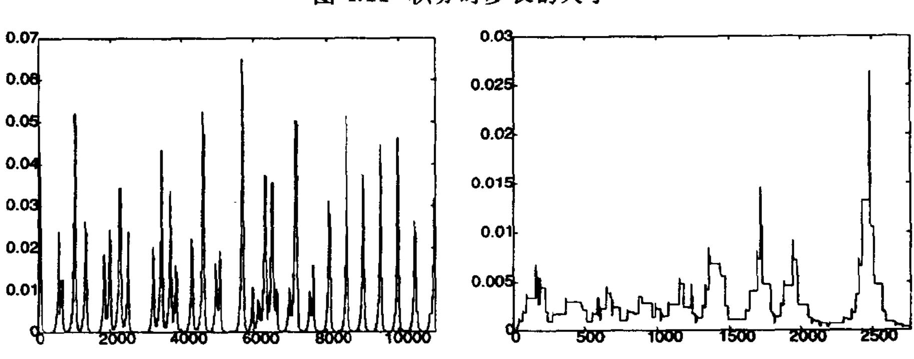
图 4.10 在 $t = 15.8299$ 附近 m_2 和 m_0 的轨道



另一个有趣的图是积分时使用的步长大小. 对于全局正则化所使用的最大步长是 0.0264 ($t = 59.4421$), 同时除了开始和最后的运动状态最小的步长是 3.7553e − 004 ($t = 52.8367$). 对于 4.2 节的直接方式最小步长是 8.8727e − 008(参见 48 页). 在图 4.11 中我们分别看到在直接方式下 (左图) 和全局正则化方式下 (右图) 所使用的步长大小. 注意这两个图尺度是不同的.

```
>> plot(diff(T1)); axis([0,11015,0,0.07])
>> plot(diff(S)); axis([0,2767,0,0.03])
```

图 4.11 积分时步长的大小



4.5 结论

实验表明，三体系统长时间的演变过程对初始数据和数值积分的精确度是极其敏感的. 典型的例子是相邻的轨道随着时间而指数地分离. 这意味着这个系统就象我们身边的一些系统(如天气预报)那样在长时间范围内是不可预测的.

然而，如果在有限的时间 $0 \le t \le t_f$ 内以足够的精确度进行数值积分，那么这个数学问题的一个确定的、唯一的解是能够被近似的.

在区间 $[0, t_f]$ 内所达到的精确度的一个粗略检验可由 Hamiltonian 函数 $K(s)$ 求得，理论上它在整个轨道上应该是零. 在我们的例子中对于 $s \in [0,8], K(s)$ 的最大值是

```
>> for k=1:n, KK(k) = K(Z(k,1:8)); end;
>> norm(KK,inf)
ans =
    1.9638e-006
```

一个更可靠的检验可以通过以不同的允许精度积分相同的轨道得到. 只要不同的近似达到足够的精确度，轨道就是有效的. 另一方面，轨道的有效性也可以由反向积分这个轨道来检验.

在 Pythagorean 三体的问题中，我们看到物体的运动是以很长的包括所有三个物体多次接近相遇的交互作用开始的. 在 $t = 46.6$ 时，最小质量的物体 m_1 穿过新形成的二元物体 m_0 和 m_2 疾驶而过，同时在第三象限被加速到几乎要脱离这个系统. 然而在 $t = 59.4$ 又受到另一个力而返了回来，此后它确定地在第一象限被排出. 另一方面二元物体又向相反的方向逃逸. 这也可能是在银河系中形成双星和晕的机制之一.

MAPLE 程序 reduce 的代码

在算法 4.4 中程序 reduce 把一个给定表达式 a 写成平方和的形式.

算法 4.4 函数 reduce

```
reduce:=proc(a) local p,P,S,c,i,j,f;
     if type(a,{name,constant}) then a
     elif type(a,'+') then p:=a; P:=convert(p,list);
        S:=map(t -> if (type(t,'^') and type(op(2,t),even))
                            then op(1,t)^(op(2,t)/2) fi, P);
        for i to nops(S) do
           for j from i+1 to nops(S) do
              if     has(p,2*S[i]*S[j]) then
                     p:=p-(S[i]^2+2*S[i]*S[j]+S[j]^2)
                         +(S[i]+S[j])^2
             elif has(p,-2*S[i]*S[j]) then
                 p:=p-(S[i]^2-2*S[i]*S[j]+S[j]^2)
                        +(S[i]-S[j])^2
             fi
          od;
       od;
```

```
          p
      else map(reduce,a)
      fi
  end:
```

参考文献

[1] C. BURRAU, *Numerische Berechnung eines Spezialfalls des Dreikorperroblems*, Astron. Nachr. 195,1913, p.113.

[2] A. GRIEWANK AND G.F. CORLISS editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, Proceedings of the SIAM Workshop on Automatic Differentiation, held in Breckenridhe,Colorado,1991, SIAM Philadelphia,1991.

[3] T. LEVI-CIVITA, *Sur la régularisation du problème des trois corps*, Acta Math. 42, 1920, pp.99-144.

[4] CH. MARCHAL, *The Three-Body Problem*, Elsevier, 1990.

[5] M.B. MONAGAN AND W.M. NEUENSCHWANDER, *GRADIENT: Algorithmic Differetiation in Maple*, Proceedings of the 1993 International Symposium on Symbolic and Algebraic Computation, ISSAC'93, 1993, p.68.

[6] C.L. SIEGEL AND J.K. MOSER, *Lectures on Celestial Mechanics*, Springer, 1971.

[7] K.F. SUNDMAN, *Memoire sur le probleme des trois corps*, Acta Math. 36,1912, pp. 105-179.

[8] V. SZEBEHELY AND C.F. PETERS, *Complete Solution of a General Problem of Three Bodies*, Astron. J. 72,1967, pp.876-883.

[9] V. SZEBEHELY, *Theory of Orbits*, Academic Press, 1967.

[10] J. WALDVOGEL, *A New Regularization of the Planar Problem of Three Bodies*, Celst. Mech. 6, 1972, pp. 221-231.

第五章　半导体的内部场

F. Klvaňa

5.1　引言

让我们考虑在 x 方向上，长度为 l 的一个半导体，它的电离活动杂质的密度为 $C(x)=C_D^+(x)-C_A^-(x)$. C_A^- 和 C_D^+ 分别是受体和施主的杂质密度，与 y 和 z 独立. 假设此半导体连接到一个具有 $U(0)=U_0$ 和 $U(l)=0$ 的外势 $U(x)$. 如果半导体在 y 和 z 方向有充分大的量纲，则所有的物理性质只依赖于 x, 我们可以把它作为一维对象来研究.

对于给定的载波 (电子和空洞) 密度 $C(x)$, 我们来考虑求半导体内势的问题. 为了简化问题，我们假设能忽略载波的复合和产生.

在标准条件下，我们可以假设电子和空洞的系统是一个经典系统. 于是，在一个内势场 $\psi(x)$ 中，由 Boltzmann 统计 ([2]) 可知电子和空洞的平衡密度 $n(x)$ 和 $p(x)$:

$$n(x)=n_i\mathrm{e}^{\frac{q}{\theta}(\psi(x)-\varphi_F)},\quad p(x)=n_i\mathrm{e}^{\frac{-q}{\theta}(\psi(x)-\varphi_F)},$$

其中 n_i 是电子的固有密度； φ_F 是所谓的 Fermi 势，在整个半导体上它是常数，并以势 $(\varphi_F=0)$ 作为参考标准； q 是一个电子的电荷； $\theta=kT$ 是统计温度.

假设上述条件成立，则内势 $\psi(x)$ 是 Poisson 方程的解，对于一维问题，它具有形式

$$\frac{\mathrm{d}^2\psi(x)}{\mathrm{d}x^2}=\frac{q}{\varepsilon}(n(x)-p(x)-C(x)),\tag{5.1}$$

其中 ε 是介质常数.

用所谓的固定势 $\psi_D(x)$ 表示密度 $C(x)$ 是有用处的，其定义为

$$C(x)=n_i(\mathrm{e}^{\frac{q}{\theta}\psi_D(x)}-\mathrm{e}^{-\frac{q}{\theta}(\psi_D(x))}).\tag{5.2}$$

方程 (5.1) 的边界条件为 Dirichlet 型，其形式为

$$\psi(0)=\psi_D(0)+U_0,\quad \psi(l)=\psi_D(l).\tag{5.3}$$

最后，用比例变换 $x=\lambda_D\cdot X$, 我们引入无量纲的量

$$\varphi(X)=\frac{q}{\theta}\psi(x),\ \varphi_D(X)=\frac{q}{\theta}\psi_D(x),\ u_0=\frac{q}{\theta}U_0,\ c(X)=\frac{C(x)}{n_i},\tag{5.4}$$

其中 $\lambda_D=\sqrt{\varepsilon\theta/q^2n_i}$ 是 Debye 长度. 由方程 (5.2) 和 (5.4) 可得到

$$\varphi_D(X)=\operatorname{arcsinh}(\frac{c(X)}{2})=\ln(\frac{1}{2}(\sqrt{c(X)^2+4}+c(X))).$$

方程 (5.1) 变成

$$\frac{\mathrm{d}^2\varphi(X)}{\mathrm{d}X^2}=\mathrm{e}^{\varphi(X)}-\mathrm{e}^{-\varphi(X)}-c(X),\tag{5.5}$$

其边界条件

$$\varphi(0)=\varphi_D(0)+u_0,\ \varphi(L)=\varphi_D(L),\ \text{其中 } L=\frac{l}{\lambda_D}.\tag{5.6}$$

5.2 利用 MAPLE 求解非线性 Poisson 方程

方程 (5.5) 和边界条件 (5.6) 描述了一个非线性边值问题，它只能数值地求解. MAPLE 不能直接解一个边值问题，为了求解，必须找到一个合适的数值方法. 我们选择有限差分法 (参见 [1]).

在区间 $[0,L]$ 上，我们定义 $N+1$ 个点 $X_0=0,X_1,\ldots,X_N=L$ 的网格，并标记 $h_i=X_{i+1}-X_i(i=0,1,\ldots,N-1)$ 和 $\varphi_i=\varphi(X_i),\varphi_{Di}=\varphi_D(X_i),c_i=c(X_i)$. 于是，方程 (5.5) 在内部点的离散化形式是

$$\frac{1}{h_i^*}\left(\frac{\varphi_{i+1}-\varphi_i}{h_i}-\frac{\varphi_i-\varphi_{i-1}}{h_{i-1}}\right)=\mathrm{e}^{\varphi_i}-\mathrm{e}^{-\varphi_i}-c_i,\quad i=1,\ldots,N-1, \tag{5.7}$$

其中 $h_i^*=\frac{1}{2}(h_{i-1}+h_i)$. 加上边界条件

$$\varphi_0=\varphi_{D0}+u_0,\ \varphi_N=\varphi_{DN},$$

成为一个未知数为 $\varphi_1,\ldots,\varphi_{N-1}$ 的非线性方程组:

$$\begin{aligned}
a_1\varphi_1+b_1\varphi_2 &= F_1(\varphi_1)\\
b_{i-1}\varphi_{i-1}+a_i\varphi_i+b_i\varphi_{i+1} &= F_i(\varphi_i),\quad \text{对于 } i=2,\ldots,N-2\\
b_{N-2}\varphi_{N-2}+a_{N-1}\varphi_{N-1} &= F_{N-1}(\varphi_{N-1})
\end{aligned}$$

其中

$$\begin{aligned}
b_i &= \tfrac{1}{h_i},\quad a_i=-(b_{i-1}+b_i),\\
F_1 &= h_1^*(\mathrm{e}^{\varphi_1}-\mathrm{e}^{-\varphi_1}-c_1)-b_0\varphi_0,\\
F_{N-1} &= h_{N-1}^*(\mathrm{e}^{\varphi_{N-1}}-\mathrm{e}^{-\varphi_{N-1}}-c_{N-1})-b_{N-1}\varphi_N,\\
F_i &= h_i^*(\mathrm{e}^{\varphi_i}-\mathrm{e}^{-\varphi_i}-c_i)\quad \text{对于 } i=2,\ldots,N-2.
\end{aligned}$$

我们把上面的方程组写成矩阵形式

$$\hat{A}\cdot\hat{\varphi}=\hat{F}(\hat{\varphi})$$

其中 $\hat{A}$ 是一个对称的三对角线矩阵，其非零元素为

$$A_{ii}=a_i,\ A_{i,i-1}=b_{i-1},\ A_{i,i+1}=b_i,$$

其中 $\hat{\varphi}$ 是分量为 $\varphi_1,\ldots,\varphi_{N-1}$ 的列向量. 我们将使用 Newton 法 (参见 [1]) 解此非线性方程组.

定义向量函数

$$\hat{G}(\hat{\varphi})=\hat{A}\cdot\hat{\varphi}-\hat{F}(\hat{\varphi}).$$

通过递推关系式

$$\hat{\varphi}^{(k+1)}=\hat{\varphi}^{(k)}+\hat{H}^{(k)},$$

Newton 法定义了系统 $\hat{G}(\hat{\varphi})=0$ 的近似解序列

$$\hat{\varphi}^{(0)},\hat{\varphi}^{(1)},\hat{\varphi}^{(2)},\ldots.$$

校正 $\hat{H}^{(k)}$ 是方程组

$$\hat{G}(\hat{\varphi}^{(k)})+\hat{J}(\hat{\varphi}^{(k)})\hat{H}^{(k)}=0$$

的解，其中 Jacobian $\hat{J}(\hat{\varphi}^{(k)})$ 包含了向量函数 $\hat{G}(\hat{\varphi}^{(k)})$ 的偏导数．系统的第 i 个方程是

$$G_i(\hat{\varphi}^{(k)})+\sum_{j=1}^{N-1}\frac{\partial G_i(\hat{\varphi}^{(k)})}{\partial\varphi_j^{(k)}}\cdot H_j^{(k)}=0 \quad \text{对于 } i=1,\ldots,N-1. \tag{5.8}$$

因为

$$\frac{\partial G_i}{\partial\varphi_j}=A_{ij}-\frac{\partial F_i}{\partial\varphi_j}=A_{ij}-h_i^*(\mathrm{e}^{\varphi_i}+\mathrm{e}^{-\varphi_i})\cdot\delta_{ij},$$

所以 Jacobian 可分成一个常数矩阵 $\hat{A}$ 和一个依赖于当前迭代的对角线矩阵 $\hat{D}(\varphi_j^{(k)})$．校正的线性方程组变成

$$(\hat{A}-\hat{D}(\varphi_j^{(k)}))\hat{H}^{(k)}=\hat{F}(\hat{\varphi}^{(k)})-\hat{A}\cdot\hat{\varphi}^{(k)}, \tag{5.9}$$

其中 $D_{ii}=h_i^*(\mathrm{e}^{\varphi_i}+\mathrm{e}^{-\varphi_i})$．因为物理的原因，建议用固定势 $\hat{\varphi}_D$ 估计初始值 $\hat{\varphi}^{(0)}$．如果

$$\|\hat{\varphi}^{(k+1)}-\hat{\varphi}^{(k)}\|_{\max}=\|\hat{H}^{(k)}\|_{\max}=\max_i|H_i^{(k)}|<\epsilon,$$

其中 ϵ 是一个允许的误差，则终止迭代过程．

证明此迭代过程的收敛性是一个复杂的任务．然而对于参数在物理上的实际数值 ([2]), Newton 法给出问题的唯一解，并且有好的收敛性．

为在 MAPLE 中解决此问题，我们取一个半导体模型，它在区域 $[0,L]$ 的中点处有 P-N 跳跃，其中 $L=10$ 及杂质分布为

$$c(X)=\tanh(20(\frac{X}{L}-\frac{1}{2})).$$

为了简化，我们选取一个步长为 $h_i=h=1/N$ 的均匀栅极，其中 N 是网格点数．用数组 phi, phiD, hp 和 Fp 分别表示 $\hat{\varphi},\hat{\varphi}_D,\hat{h}^*$ 和 $\hat{F}$．三对角线矩阵将由对角线和非对角线向量 A,Ap 和 B 表示，其中

$$A_i=a_i,\quad Ap_i=a_i-h_i^*(\mathrm{e}^{\varphi_i}+\mathrm{e}^{-\varphi_i}),\ B_i=b_i.$$

MAPLE 程序的第一部分是：

```
>      #dimensionless formulation, x,L in units of Debye's length
> N:=20:  L:=10.0:  h:=L/N:
> U0:=0:  #normalized potential on the boundary
>      #concentration of impurities - region of N-P jump
> c:=tanh(20.0*(x/L-0.5)):
>      #mesh of N+1 point for region 0..L
> xp:=array(0..N):
> for i from 0 to N do xp[i]:=i*h od:
>      #array of concentrations Ca and builtin potential phiD
> phiD:=array(0..N):
> Ca:=array(0..N):
> for i from 0 to N do
>     Ca[i]:=evalf(subs(x=xp[i],c));
>     phiD[i]:=arcsinh(Ca[i]/2)
> od:
```

为了求解三对角线系统 (5.9), 我们用标准库函数 linsolve, 但由于没有考虑矩阵的稀疏性，它运行的较慢．因此，我们对三对角线系统实行没有转轴的标准 LU 分解，它运行很快，且所需存储量

少. 函数 solvetridiag(n, A, B, F) 求解 n 个线性方程的对称三对角线系统; A 是主对角线向量, B 包含非对角线元素, F 是右端的向量. 此函数返回的解为数组 [0..n+1], 下标 0 和 $n+1$ 是向量 F 对应的分量 (参见算法 (5.1)).

算法 5.1 MAPLE 函数 solvetridiag 和 norma

```
solvetridiag:=proc(N, A, B, F) local alfa, beta, pom, x, i;
    # solution of a linear system of  N equations
    # with tridiagonal symmetric matrix, where
    # A is a vector diagonal of elements,
    # B is a vector of off-diagonal elements and
    # F is a vector of right hand site,
    #result is a vector (0..N+1)
    x:=array(0..N+1,[(0)=F[0],(N+1)=F[N+1]]);
    alfa[1]:= -B[1]/A[1]; beta[1]:= F[1]/A[1];
    for i from 2 to N-1 do
        pom:= A[i]+B[i-1]*alfa[i-1];
        alfa[i]:= -B[i]/pom;
        beta[i]:= (F[i]-B[i-1]*beta[i-1])/pom
    od;
    x[N]:= (F[N]-B[N-1]*beta[N-1])/(A[N]+B[N-1]*alfa[N-1]);
    for i from N-1 by -1 to 1 do
        x[i]:= alfa[i]*x[i+1]+beta[i]
    od;
    eval(x)
end;  #solvetridiag

norma:= proc(N, X) local mx,i,pom;
    #calculation of a infinity norm for 1-D array X, i=1..N
    mx:=0;
    for i to N do
         pom:=abs(X[i]);
         if pom > mx then mx := pom fi
    od;
    mx
end;
```

现在实现 Newton 法就十分简单了. 我们定义函数 iterate(), 它执行迭代, 从老的势计算新的势 phi, 直到校正 H 的范数小于允许值 epsilon. iterate() 也返回最后两步迭代之差的范数. 我们取 epsilon= 0.001.

算法 5.2 定义迭代函数

```
iterate:=proc(phi) local i, nor, H, ex, Ap, Fp;
```

```
    #iterations for phi begin
    nor:=1;
    while nor > epsilon do
       for i to N-1 do
          ex := exp(phi[i]);
          Ap[i] := A[i] - hp[i]*(ex+1/ex);
          Fp[i] := Fpoc[i] + hp[i]*(ex-1/ex) - A[i]*phi[i];
          if i = 1 then Fp[1] := Fp[1] - B[1]*phi[2]
            elif i = N-1 then
              Fp[N-1] := Fp[N-1] - B[N-2]*phi[N-2]
            else
              Fp[i] := Fp[i] - B[i-1]*phi[i-1] - B[i]*phi[i+1]
          fi;
       od;
       H:=solvetridiag(N-1,Ap,B,Fp);
       for i to N-1 do
         phi[i] := H[i] + phi[i];
         nor := norma(N-1,H);
       od;
    od;
 end; #iterate

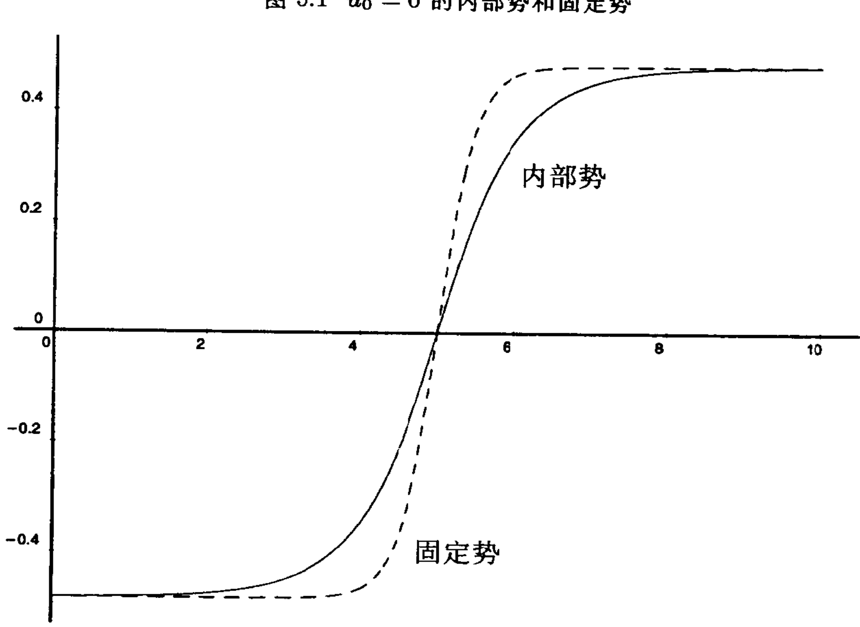
>     #generation of vectors A,B and average step hp
> B := array(0..N-1):  A := array(1..N): hp := array(1..N-1):
> for i from 0 to N-1 do B[i] := 1/(xp[i+1]-xp[i]) od:
> for i from 1 to N-1 do A[i] := -(B[i-1]+B[i]) od:
> for i from 1 to N-1 do hp[i] := (xp[i+1]-xp[i-1])/2 od:
> phi := array(0..N):
>     #initial approximation of phi
> phi := eval(phiD):
>     #introducing boundary condition
> phi[0] := phi[0]+U0:
>     #generation of vectors of right hand side Fpoc and Fp
> Fpoc := array(0..N): Fp := array(0..N):
> for i to N-1 do Fpoc[i] := -hp[i]*Ca[i] od:
> Fpoc[1] := Fpoc[1] - B[0]*phi[0]:
> Fpoc[N-1] := Fpoc[N-1] - B[N-1]*phi[N]:
>     #introducing the boundary conditions into Fp
> Fp[0] := phi[0]:  Fp[N] := phi[N]:
> epsilon := 0.001:  #error of the result
>     #solution for phi
> iterate(phi); #returns a value of norm for iteruj:=proc(phi)
```

$$.1899345025\ 10^{-6}$$

```
>       #graph of builtin potential phiD
>  GphiD:=plot(arcsinh(c/2),x=0..L, color=black, linestyle=2):
>       # plot of resulting phi and phiD
>  Gphi:= plot([seq([xp[i],phi[i]],i=0..N)],color=black):
>  plots[display]({Gphi,GphiD}); #end
```

内部势和固定势的图形如图 5.1.

图 5.1 $u_0 = 0$ 的内部势和固定势



5.3 MATLAB 解法

在前一节里, 利用问题的特殊性质 (如系统 (5.9) 的矩阵的三对角线性), 给出求解一个纯数值问题的比较快的 MAPLE 算法. 当然, 象 MATLAB 这样的数值系统更适合求解纯数值问题. 在 MATLAB 中, 为了处理稀疏矩阵, 我们利用标准结构, 可以给出解决问题的一个非常简单的程序.

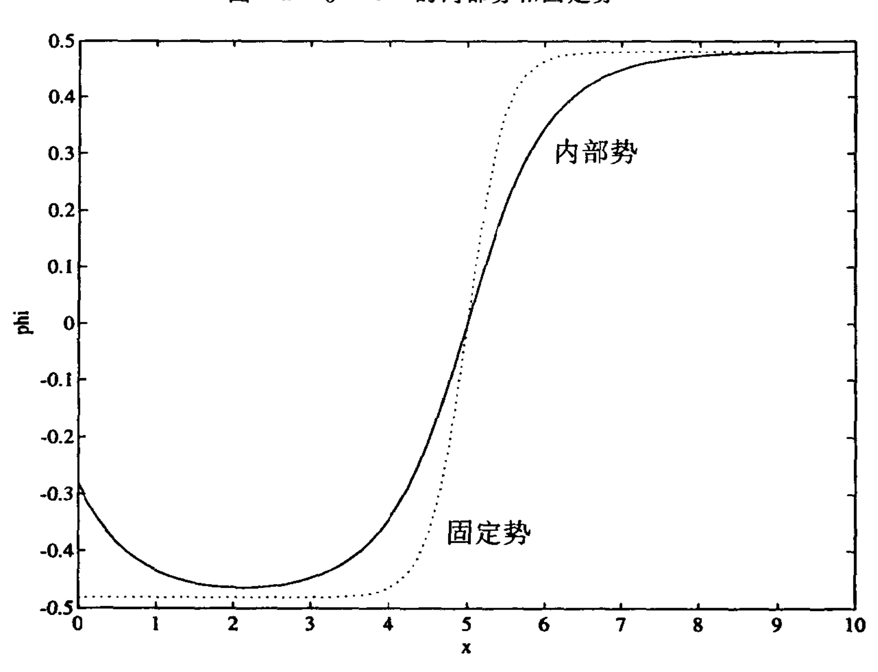
MATLAB 可用函数 spdiags 定义稀疏矩阵. 标准的矩阵运算能非常有效地处理这样的稀疏矩阵. 这样可用标准的符号表示线性系统 $A \cdot x = f$ 的解:

```
>> x=A\f
```

我们再用 Newton 法, 求解非线性 Poisson 方程 (5.7) 的离散化形式. 用函数 spdiagsx 定义三对角线矩阵 $\hat{A}$ 和对角线矩阵 $\hat{D}$. MATLAB 程序的结果是比较直观的:

```
%program for computing the internal potential in a semiconductor
l = 10; n = 80;
u0 = 0.2                        %external potential
xl = (0:1/n:1)'                 %mesh
cl = tanh(20.0.*(xl./1-0.5)) %concentration of impurities
```

图 5.2 $u_0 = 0.2$ 的内部势和固定势




```
phid = asinh(cl*0.5)           %builtin potential
h = xl(2:n+1) - xl(1:n)        % steps
b = 1 ./ h
                               %tridiagonal matrix A
A=spdiags([b(2:n)-(b(1:n-1)+b(2:n)) b(1:n-1)],-1:1,n-1,n-1)

hs = (h(1:n-1)+h(2:n))*0.5     %average steps
c = cl(2:n)

f0 = -hs.*c
f0(1) = f0(1)-(phid(1)+u0)*b(1);
f0(n-1) = f0(n-1)-phid(n+1)*b(n);
phi = phid(2:n)                % initial approximation

nor = 1                        % iteration
while nor > 0.001
  ex=exp(phi); f=f0+hs.*(ex-1./ex)-A*phi
  Dp=spdiags(hs.*(ex+1./ex),0,n-1,n-1)
  H=(A-Dp)\f
  nor = norm(H,inf)
  phi = phi + H
```

```
end
phiv=[phid(1)+u0;phi;phid(n+1)] %with boundary conditions

clf;
plot(xl,phiv,'-r',xl,phid,':g')
hold;
text(4.6,-0.4,'Builtin potential');
text(6,0.3,'Internal potential');
xlabel('x'); ylabel('phi');
                                    % end
```

当外势 $u_0 = 0.2$ 时，此结果的图形由图 5.2 给出.

参考文献

[1] W.H. PRESS, B.P. FLANNERY, S.A. TEUKOLSKY and W.T. VETTERLING, *Numerical Recipes*, Cambridge University Press, 1988.

[2] S. SELBERHERR, *Analysis and Simulation of Semiconductor Devices*, Springer, 1984.

第六章　某些最小二乘问题

W. Gander and U. von Matt

6.1　引言

这一章我们将讨论某些最小二乘问题，它是在使用坐标测量技术的制造业的质量控制问题中出现的 [4][5]. 对于大批量生产，机器生产的是零件，重要的是要了解生产的零件是否满足质量要求. 因此，通常需要从生产线上抽取一些样本零件进行测量同时与标准件进行比较. 如果它们不满足特定的公差，这机器可能必须维修或调整以生产更好的零件.

6.2　拟合平面上的直线、矩形和正方形

用直线去拟合一组点以最小化已知点到这条直线的距离的平方和，这个问题来源于惯性张量主轴的计算问题. 在 [1] 中这个事实被用于以直线或平面拟合三维空间点集的问题，它是通过求解 3×3 矩阵的特征值问题来解决的.

在这一节我们将研究基于奇异值分解的不同的算法，它将允许我们用直线、矩形和正方形拟合平面上的一组观测点，对于三维空间的某些问题它也同样有效.

我们首先考虑以直线拟合平面上的一组已知点 $P_1, P_2, \dots P_m$ 的问题. 我们用 $(x_{P_1}, y_{P1}), (x_{P_2}, y_{P_2}), \dots, (x_{P_m}, y_{P_m})$ 表示它们的坐标. 有时我们把所有点的 x 和 y 坐标表示为向量是有用的. 我们将使用 $\mathbf{x}_P$ 表示向量 $(x_{P_1}, x_{P_2}, \dots x_{P_m})$, 类似地用 $\mathbf{y}_P$ 表示 y 坐标.

我们要解决的问题不是线性回归问题. 线性回归意味着用线性模型 $y = ax + b$ 拟合已知点，也就是说要确定两个参数 a 和 b 使得剩余平方和最小:

$$\sum_{i=1}^{n} r_i^2 = \min, \qquad \text{其中} \qquad r_i = y_{P_i} - a x_{P_i} - b.$$

这个简单的线性最小二乘问题在 MATLAB 中可使用下面的语句求解 (假设 $\mathbf{x} = \mathbf{x}_P$ 和 $\mathbf{y} = \mathbf{y}_P$)

```
>> p = [ x ones(size(x))]\y;
>> a = p(1); b = p(2);
```

在线性回归的情形下，点 P_i 到所拟合线性函数的 y 坐标之差的平方和被极小化了. 然而现在我们要极小化的是 *这些点到所拟合直线的距离的平方和*.

在平面上我们能够用如下方程唯一地表示一条直线

$$c + n_1 x + n_2 y = 0, \qquad n_1^2 + n_2^2 = 1. \tag{6.1}$$

单位向量 (n_1, n_2) 正交于这条直线. 一个点在这条直线上当且仅当它的坐标 (x, y) 满足第一个方程. 另一方面，如果 $P = (x_P, y_P)$ 是不在直线上的点，我们计算

$$r = c + n_1 x_P + n_2 y_P,$$

则 $|r|$ 是它到这条直线的距离. 因此如果我们想要确定一条直线，使给定点到它距离的平方和最小，我们必须求解一个约束的最小二乘问题

$$\|\mathbf{r}\| = \sum_{i=1}^{m} r_i^2 = \min$$

满足

$$\begin{pmatrix} 1 & x_{P_1} & y_{P_1} \\ 1 & x_{P_2} & y_{P_2} \\ \vdots & \vdots & \vdots \\ 1 & x_{P_m} & y_{P_m} \end{pmatrix} \begin{pmatrix} c \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix} \quad \text{和} \quad n_1^2 + n_2^2 = 1. \tag{6.2}$$

令 $\mathbf{A}$ 是线性方程组 (6.2) 的矩阵， $\mathbf{x}$ 表示未知向量 $(c, n_1, n_2)^{\mathrm{T}}$, 同时 $\mathbf{r}$ 是方程的右端. 因为正交变换 $\mathbf{y} = \mathbf{Q}^{\mathrm{T}}\mathbf{r}$ 保持范数不变 (对于正交矩阵 $\mathbf{Q}$, $\|\mathbf{y}\|_2 = \|\mathbf{r}\|_2$), 我们可以按如下所述去解决问题 (6.2).

首先我们计算 $\mathbf{A}$ 的 QR 分解，同时把我们的问题简化为求解一个小的方程组:

$$\mathbf{A} = \mathbf{Q}\mathbf{R} \quad \Rightarrow \quad \mathbf{Q}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} c \\ n_1 \\ n_2 \end{pmatrix} = \mathbf{Q}^{\mathrm{T}}\mathbf{r} \tag{6.3}$$

因为非线性约束仅仅包含两个未知量，现在我们需要求解

$$\begin{pmatrix} r_{22} & r_{23} \\ 0 & r_{33} \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \text{满足} \quad n_1^2 + n_2^2 = 1. \tag{6.4}$$

算法 6.1 函数 clsq

```
function [c,n] = clsq(A,dim);
% solves the constrained least squares Problem
% A (c n)' ~ 0 subject to norm(n,2)=1
% length(n) = dim
% [c,n] = clsq(A,dim)
[m,p] = size(A);
if p < dim+1, error ('not enough unknowns'); end;
if m < dim, error ('not enough equations'); end;
m = min (m, p);
R = triu (qr (A));
[U,S,V] = svd(R(p-dim+1:m,p-dim+1:p));
n = V(:,dim);
c = -R(1:p-dim,1:p-dim)\R(1:p-dim,p-dim+1:p)*n;
```

问题 (6.4) 是形如 $\|\mathbf{Bx}\|_2 = \min$ 并满足 $\|\mathbf{x}\|_2 = 1$ 的问题. 极小值是 $\mathbf{B}$ 的最小奇异值, 同时问题的解可由相应的奇异向量给出 [2]. 于是我们可以由 2×2 矩阵的奇异值分解来确定 n_1 和 n_2. 把这些值代入方程组 (6.3) 的第一个分量同时令它等于零, 我们就能计算出 c. 稍作推广, 我们用 dim 表示法向量 $\mathbf{n}$ 的维数, 则用于求解问题 (6.2) 的 MATLAB 函数由算法 6.1 给出. 我们用如下的主程序检验函数 clsq:

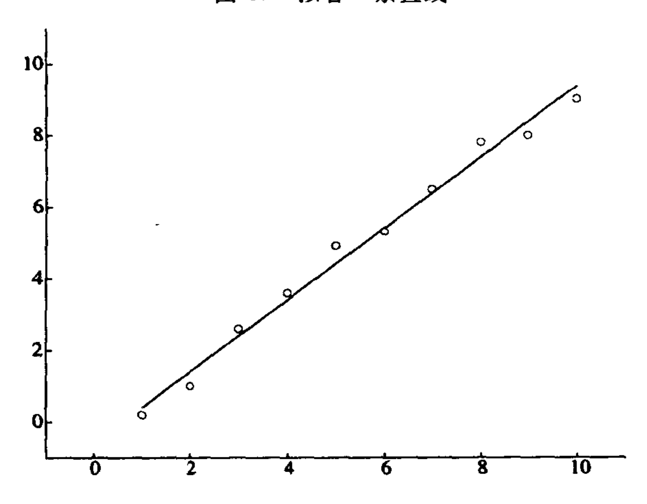
```
>> % mainline.m
>> Px = [1:10]'
>> Py = [ 0.2 1.0 2.6 3.6 4.9 5.3 6.5 7.8 8.0 9.0]'
>> A = [ones(size(Px)) Px Py]
>> [c, n] = clsq(A,2)
```

由程序 mainline 计算出的直线为: $0.4162 - 0.7057x + 0.7086y = 0$. 我们现在要画出这些点和拟合的直线. 为此我们需要由算法 6.2 给出的函数 ployline. 图形由如下附加于程序 mainline 的命令给出. 结果如图 6.1.

```
>> clf; hold on;
>> axis([-1, 11 -1, 11])
>> plotline(Px,Py,'o',c,n,'-')
>> hold off;
```



图 6.1 拟合一条直线



算法 6.2 函数 plotline

```
function plotline(x,y,s,c,n,t)
% plots the set of points (x,y) using the symbol s
% and plots the straight line c+n1*x+n2*y=0 using
% the line type defined by t
plot(x,y,s)
xrange = [min(x) max(x)];
yrange = [min(y) max(y)];
```

```
if n(1)==0, % c+n2*y=0  => y = -c/n(2)
   x1=xrange(1); y1 = -c/n(2);
   x2=xrange(2); y2 = y1
elseif n(2) == 0, % c+n1*x=0  => x = -c/n(1)
   y1=yrange(1); x1 = -c/n(1);
   y2=yrange(2); x2 = x1;
elseif xrange(2)-xrange(1)> yrange(2)-yrange(1),
   x1=xrange(1); y1 = -(c+n(1)*x1)/n(2);
   x2=xrange(2); y2 = -(c+n(1)*x2)/n(2);
else
   y1=yrange(1); x1 = -(c+n(2)*y1)/n(1);
   y2=yrange(2); x2 = -(c+n(2)*y2)/n(1);
end
plot([x1, x2], [y1,y2],t)
```

拟合两条平行线

要拟合两条平行线，我们需要两组点．我们用 $\{P_i\}, i=1,\dots,p$ 和 $\{Q_j\}, j=1,\dots,q$ 表示这两组点．因为这两条线是平行的，它们的法线向量必须是相同的．于是这两条线的方程是

$$\begin{aligned} c_1 + n_1 x + n_2 y &= 0, \\ c_2 + n_1 x + n_2 y &= 0, \\ n_1^2 + n_2^2 &= 1. \end{aligned}$$

如果我们把两组点的坐标代入这些方程我们就得到了如下的约束最小二乘问题：

$$\|\mathbf{r}\| = \sum_{i=1}^{m} r_i^2 = \min$$

满足

$$\begin{pmatrix} 1 & 0 & x_{P_1} & y_{P_1} \\ 1 & 0 & x_{P_2} & y_{P_2} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & x_{P_p} & y_{P_p} \\ 0 & 1 & x_{Q_1} & y_{Q_1} \\ 0 & 1 & x_{Q_2} & y_{Q_2} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & x_{Q_q} & y_{Q_q} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{p+q} \end{pmatrix} \quad \text{和} \quad n_1^2 + n_2^2 = 1. \tag{6.5}$$

我们可以再一次使用函数 clsq 解决这个问题：

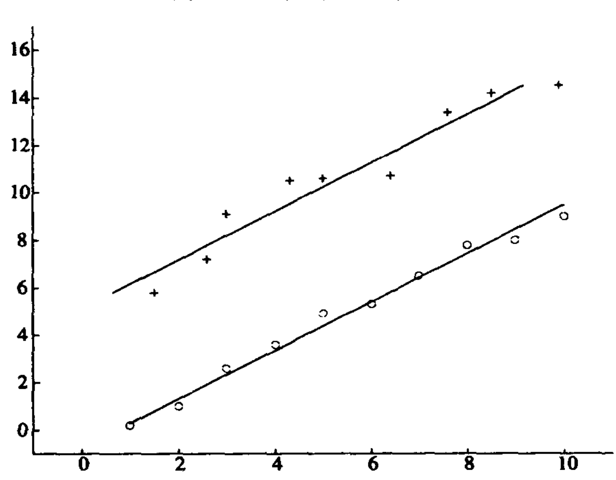
```
>> % mainparallel.m
>> Px = [1:10]'
>> Py = [ 0.2 1.0 2.6 3.6 4.9 5.3 6.5 7.8 8.0 9.0]'
>> Qx = [ 1.5 2.6 3.0 4.3 5.0 6.4 7.6 8.5 9.9 ]'
```

```
>> Qy = [ 5.8 7.2 9.1 10.5 10.6 10.7 13.4 14.2 14.5]'
>> A = [ones(size(Px))  zeros(size(Px)) Px Py
>>     zeros(size(Qx))  ones(size(Qx)) Qx Qy ]
>> [c, n] = clsq(A,2)
>> clf; hold on;
>> axis([-1 11 -1 17])
>> plotline(Px,Py,'o',c(1),n,'-')
>> plotline(Qx,Qy,'+',c(2),n,'-')
>> hold off;
```

由程序 mainparallel 得到的结果是两条直线，如图 6.2.

$$\begin{aligned} 0.5091 - 0.7146x + 0.6996y &= 0, \\ -3.5877 - 0.7146x + 0.6996y &= 0. \end{aligned}$$

图 6.2 拟合两条平行的直线



拟合垂直的直线

为了拟合两条垂直的直线，我们可以按照类似于平行线的情形来做. 如果 (n_1, n_2) 是第一条直线的法线向量，则由于正交性第二条直线一定有法线向量 $(-n_2, n_1)$. 因此我们仍然有四个未知参数 c_1, c_2, n_1, n_2. 如果 P_i 是与第一条线相关联的点， Q_j 是与第二条线相关联的点，我们就得到了如下的约束最小二乘问题：

$$\|\mathbf{r}\| = \sum_{i=1}^{m} r_i^2 = \min$$

满足

$$\begin{pmatrix} 1 & 0 & x_{P_1} & y_{P_1} \\ 1 & 0 & x_{P_2} & y_{P_2} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & x_{P_p} & y_{P_p} \\ 0 & 1 & y_{Q_1} & -x_{Q_1} \\ 0 & 1 & y_{Q_2} & -x_{Q_2} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & y_{Q_q} & -x_{Q_q} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{p+q} \end{pmatrix} \quad \text{和} \quad n_1^2 + n_2^2 = 1. \tag{6.6}$$

为了计算两条垂直直线的方程，我们仅仅需要改变在 mainparallel 中矩阵 A 的定义. 为了得到更好的绘图对于第二组点我们同样选择不同的数值.

```
>> % mainorthogonal.m
>> Px = [1:10]'
>> Py = [ 0.2 1.0 2.6 3.6 4.9 5.3 6.5 7.8 8.0 9.0]'
>> Qx = [ 0 1 3 5 6 7]'
>> Qy = [12 8 6 3 3 0]'
>> A = [ones(size(Px))   zeros(size(Px)) Px  Py
>>      zeros(size(Qx))  ones(size(Qx))  Qy -Qx  ]
>> [c, n] = clsq(A,2)
>> clf; hold on;
>> axis([-1 11 -1 13])
>> axis('equal')
>> plotline(Px,Py,'o',c(1),n,'-')
>> n2(1) =-n(2); n2(2) = n(1)
>> plotline(Qx,Qy,'+',c(2),n2,'-')
```

程序 mainorthogonal 计算出两条垂直的直线

$$\begin{aligned} -0.2527 - 0.6384x + 0.7697y &= 0, \\ 6.2271 - 0.7697x - 0.6384y &= 0. \end{aligned}$$

我们同样能够对每一组点独立地拟合两条直线:

```
>> [c, n] = clsq([ones(size(Px)) Px Py],2)
>> plotline(Px,Py,'+',c,n,':')
>> [c, n] = clsq([ones(size(Qx)) Qx Qy],2)
>> plotline(Qx,Qy,'+',c,n,':')
>> hold off;
```

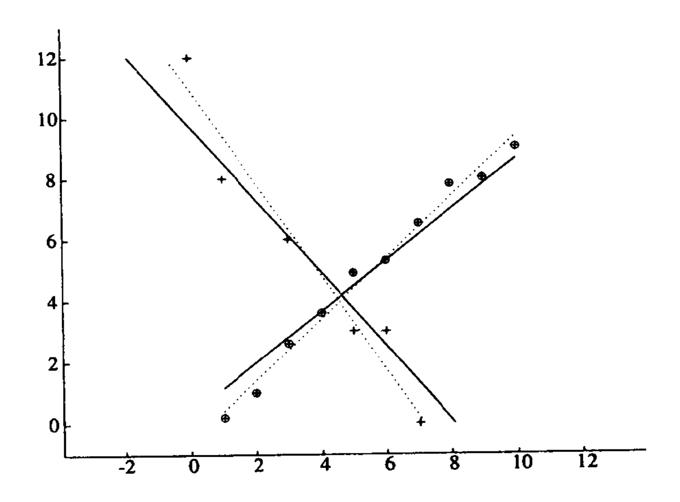
独立计算出来的直线在图 6.3 中用点线来表示.

拟合一个矩形

拟合一个矩形需要四组数据

$$P_i, i = 1, \ldots, p, \quad Q_j, j = 1, \ldots, q, \quad R_k, k = 1, \ldots, r, \quad S_l, l = 1, \ldots, s.$$

图 6.3 拟合两条垂直的直线



因为矩形的边是相互平行或垂直的，我们可以非常相似于前面来进行．四个边将有方程

$$\begin{aligned} a: \quad c_1 + n_1 x + n_2 y &= 0 \\ b: \quad c_2 - n_2 x + n_1 y &= 0 \\ c: \quad c_3 + n_1 x + n_2 y &= 0 \\ d: \quad c_4 - n_2 x + n_1 y &= 0 \\ n_1^2 + n_2^2 &= 1. \end{aligned}$$

代入四组点我们得到如下的约束最小二乘问题

$$\|\mathbf{r}\| = \sum_{i=1}^{m} r_i^2 = \min$$

满足

$$\begin{pmatrix} 1 & 0 & 0 & 0 & x_{P_1} & y_{P_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & x_{P_p} & y_{P_p} \\ 0 & 1 & 0 & 0 & y_{Q_1} & -x_{Q_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & y_{Q_q} & -x_{Q_q} \\ 0 & 0 & 1 & 0 & x_{R_1} & y_{R_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & x_{R_r} & y_{R_r} \\ 0 & 0 & 0 & 1 & y_{S_1} & -x_{S_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & y_{S_s} & -x_{S_s} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{p+q+r+s} \end{pmatrix} \quad 和 \quad n_1^2 + n_2^2 = 1. \qquad (6.7)$$

我们不再明确地给定四组点的坐标，而是使用 MATLAB 中的函数 ginput 用鼠标输入这些点. [X,Y]=ginput(N) 从当前的坐标下得到这 N 个点同时以长度为 N 的向量 X 和 Y 返回 x 坐标和 y 坐标. 这些点必须按顺时针方向或反时针方向来输入，其次序要与矩形的边的顺序相同：下一个边总是正交于前一个边.

```
>> % rectangle.m
>> clf; hold on;
>> axis([0 10 0 10])
>> axis('equal')
>> p=100; q=100; r=100; s=100;

>> disp('enter points P_i belonging to side A')
>> disp('by clicking the mouse in the graphical window.')
>> disp('Finish the input by pressing the Return key')
>> [Px,Py] = ginput(p); plot(Px,Py,'o')
>> disp('enter points Q_i for side B ')
>> [Qx,Qy] = ginput(q); plot(Qx,Qy,'x')
>> disp('enter points R_i for side C ')
>> [Rx,Ry] = ginput(r); plot(Rx,Ry,'*')
>> disp('enter points S_i for side D ')
>> [Sx,Sy] = ginput(s); plot(Sx,Sy,'+')

>> zp = zeros(size(Px)); op =  ones(size(Px));
>> zq = zeros(size(Qx)); oq =  ones(size(Qx));
>> zr = zeros(size(Rx)); or =  ones(size(Rx));
>> zs = zeros(size(Sx)); os =  ones(size(Sx));

>> A = [ op zp zp zp Px  Py
>>       zq oq zq zq Qy -Qx
>>       zr zr or zr Rx  Ry
>>       zs zs zs os Sy -Sx]

>> [c, n] = clsq(A,2)

>> % compute the 4 corners of the rectangle
>> B  = [n  [-n(2) n(1)]']
>> X = -B* [c([1 3 3 1])'; c([2 2 4 4])']
>> X = [X X(:,1)]
>> plot(X(1,:), X(2,:))

>> % compute the individual lines, if possible
```

```
>> if all([sum(op)>1 sum(oq)>1 sum(or)>1 sum(os)>1]),
>>    [c1, n1] = clsq([op Px Py],2)
>>    [c2, n2] = clsq([oq Qx Qy],2)
>>    [c3, n3] = clsq([or Rx Ry],2)
>>    [c4, n4] = clsq([os Sx Sy],2)

>>    % and their intersection points
>>    aaa = -[n1(1) n1(2); n2(1) n2(2)]\[c1; c2];
>>    bbb = -[n2(1) n2(2); n3(1) n3(2)]\[c2; c3];
>>    ccc = -[n3(1) n3(2); n4(1) n4(2)]\[c3; c4];
>>    ddd = -[n4(1) n4(2); n1(1) n1(2)]\[c4; c1];

>>    plot([aaa(1) bbb(1) ccc(1) ddd(1) aaa(1)], ...
>>         [aaa(2) bbb(2) ccc(2) ddd(2) aaa(2)],':')
>> end
>> hold off;
```

程序 rectangle 不仅可以计算矩形，而且同样可以对每一组点拟合单独的直线．结果都显示于图 6.4 中．为理解某些语句，加一些注解是必要的．为求出矩阵顶点的坐标，我们计算相应边直线的交点．我们需要求解线性方程组

$$\begin{array}{rcl} n_1x+n_2y &=& -c_1 \\ -n_2x+n_1y &=& -c_2 \end{array} \iff C\mathbf{x}=-\begin{pmatrix} c_1 \\ c_2 \end{pmatrix},\quad C=\begin{pmatrix} n_1 & n_2 \\ -n_2 & n_1 \end{pmatrix}$$

因为 C 是正交的，我们能够简单地用 $B=C^{\mathrm{T}}$ 乘以方程的右端得到所要的解．对于 4 个顶点只要适当的安排方程，使得矩阵 C 总是方程组的系数矩阵，用如下简捷的语句我们就能够同时计算所有 4 个顶点的坐标.

```
>> X = -B* [c([1 3 3 1])'; c([2 2 4 4])'].
```

拟合正方形

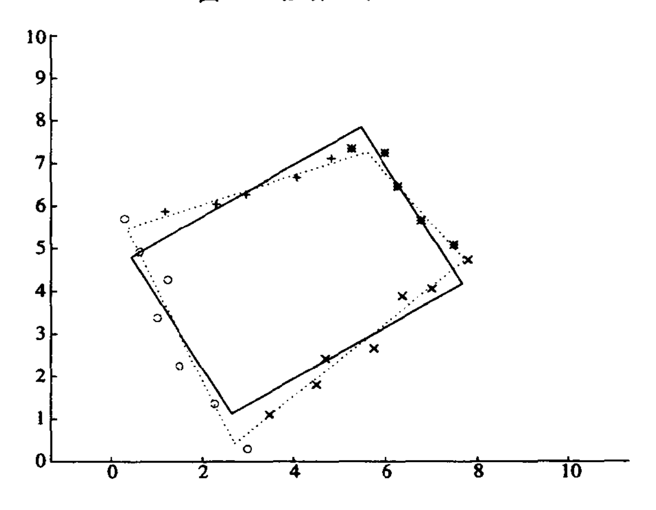
如果用 $|d|$ 表示正方形的边的长度，则按照反时针的方向四个边有方程

$$\begin{array}{lrcl} \alpha: & c_1+n_1x+n_2y &=& 0 \\ \beta: & c_2-n_2x+n_1y &=& 0 \\ \gamma: & d+c_1+n_1x+n_2y &=& 0 \\ \delta: & d+c_2-n_2x+n_1y &=& 0 \\ & n_1^2+n_2^2 &=& 1. \end{array} \tag{6.8}$$

四个边的走向是重要的：在第三和第四个方程中的参数 d 应有相同的符号．如果我们选择顺时针方向，则直线 β 和 δ 方程的法向量必须改变符号，即我们必须使用方程

$$\begin{array}{lrcl} \beta: & c_2+n_2x-n_1y &=& 0, \\ \delta: & d+c_2+n_2x-n_1y &=& 0. \end{array}$$

图 6.4 拟合一个矩形



下面我们假设按反时针方向的四个边的四组点为

$$P_i, i=1,\dots,p, \quad Q_j, j=1,\dots,q, \quad R_k, k=1,\dots,r, \quad S_l, l=1,\dots,s.$$

则我们的问题就变为

$$\|\mathbf{r}\| = \sum_{i=1}^{m} r_i^2 = \min$$

满足

$$\begin{pmatrix} 0 & 1 & 0 & x_{P_1} & y_{P_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & x_{P_p} & y_{P_p} \\ 0 & 0 & 1 & y_{Q_1} & -x_{Q_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & y_{Q_q} & -x_{Q_q} \\ 1 & 1 & 0 & x_{R_1} & y_{R_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 0 & x_{R_r} & y_{R_r} \\ 1 & 0 & 1 & y_{S_1} & -x_{S_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 1 & y_{S_s} & -x_{S_s} \end{pmatrix} \begin{pmatrix} d \\ c_1 \\ c_2 \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{p+q+r+s} \end{pmatrix} \quad 和 \quad n_1^2 + n_2^2 = 1. \tag{6.9}$$

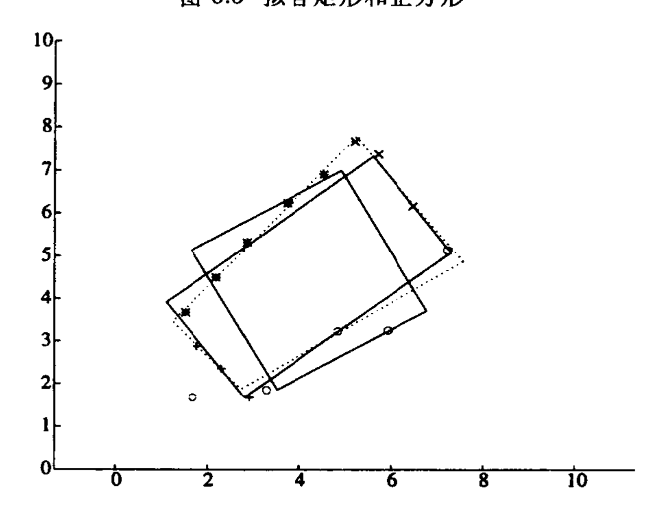
我们可以重新使用程序 `rectangle` 的大部分. 变化仅仅在于为绘制正方形重新定义矩阵 A 和向量 c. 在输入这些点时我们把正方形的边按反时针方向排列.

```
>> A = [ zp op zp  Px  Py
>>       zq zq oq  Qy -Qx
>>       or or zr  Rx  Ry
```

```
>>       os zs os  Sy -Sx]
>> [cc, n] = clsq(A,2)
>> % compute the vector of constants for the four lines
>> c(1) = cc(2); c(2) = cc(3); c(3) = c(1)+cc(1);
>> c(4) = c(2)+cc(1);  c=c(:);
>> % compute the 4 corners of the square
>> B  = [n  [-n(2) n(1)]']
>> X = -B* [c([1 3 3 1])'; c([2 2 4 4])']
>> X = [X X(:,1)]
>> plot(X(1,:), X(2,:))
>> hold off;
```

在图 6.5 中给出了最优的矩形和拟合最优的正方形. 注意到在极端的情况下, 我们可能得不到我们希望的结果. 参看图 6.6, 这里对于四个点 $P=(2.5,1.5), Q=(6.5,4), R=(7,9), S=(2,8)$ 我们计算出 "最优的正方形". 似乎这个正方形不是最优的拟合. 然而, 如果注意到点 P 属于第一条边 a, 点 Q 属于 b, 点 R 属于 c 同时 S 属于 d. 这四个点刚好位于这些边所在的直线上, 因此它们的距离的平方和是零!

图 6.5 拟合矩形和正方形



6.3 拟合超平面

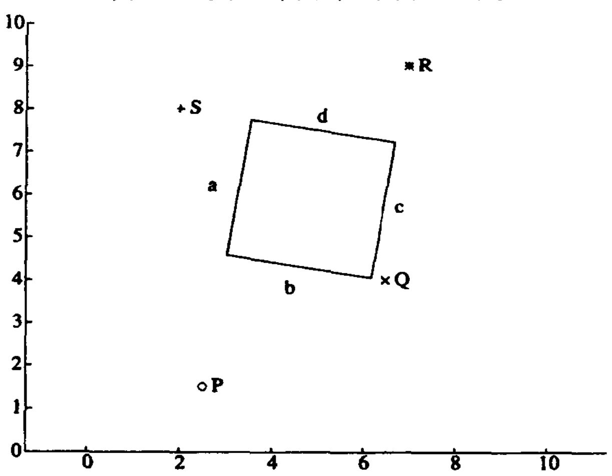
函数 clsq 可以用于对 R^n 中的点拟合 $(n-1)$ 维的超平面. 令矩阵 $X=[\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_m]^{\mathrm{T}}$ 的行表示已知点的坐标, 即点 P_i 有坐标 $\mathbf{x}_i=X(i,:), i=1,\ldots,m$. 则调用

```
>> [c, N] = clsq ([ones(m,1) X], n);
```

就可以确定正规形式的超平面 $c+N_1y_1+N_2y_2+\cdots+N_ny_n=0$.

在这一节中, 我们将说明如何计算较低的 s 维超平面的最佳拟合, 其中 $1\le s\le n-1$. 我们

图 6.6 对于四个点的“最优”正方形



将遵循 [3] 所发展的理论. R^n 中的一个 s 维超平面 α 可以被表示为参数形式:

$$\alpha: \quad \mathbf{y} = \mathbf{p} + \mathbf{a}_1 t_1 + \mathbf{a}_2 t_2 + \cdots + \mathbf{a}_s t_s = \mathbf{p} + A\mathbf{t}. \tag{6.10}$$

在这个方程中 $\mathbf{p}$ 是平面上的一个点, 同时 $\mathbf{a}_i$ 是线性无关的方向向量, 于是这个超平面是由 $\mathbf{p}$ 和 $A = [\mathbf{a}_1, \ldots, \mathbf{a}_s]$ 确定的.

不失一般性, 我们设 A 是正交的, 即 $A^{\mathrm{T}}A = I_s$. 如果我们想对已知的一组点 X 拟合一个超平面, 则我们必须极小化点到平面的距离. 点 $P_i = \mathbf{x}_i$ 到超平面的距离 d_i 是

$$d_i = \min_{\mathbf{t}} \|\mathbf{p} - \mathbf{x}_i + A\mathbf{t}\|_2.$$

为了求得极小值我们关于 $\mathbf{t}$ 求解 $\operatorname{grad} d_i^2 = 2A^{\mathrm{T}}(\mathbf{p} - \mathbf{x}_i + A\mathbf{t}) = \mathbf{0}$, 同时因为 A 是正交的, 我们得到

$$\mathbf{t} = A^{\mathrm{T}}(\mathbf{x}_i - \mathbf{p}). \tag{6.11}$$

因此这个距离变为

$$d_i^2 = \|\mathbf{p} - \mathbf{x}_i + AA^{\mathrm{T}}(\mathbf{x}_i - \mathbf{p})\|_2^2 = \|\mathcal{P}(\mathbf{x}_i - \mathbf{p})\|_2^2,$$

其中 $\mathcal{P} = I - AA^{\mathrm{T}}$, 它是到 A 的余集上的投影, 即 A^{T} 的零空间.

我们的目的是极小化所有点到超平面的距离的平方和. 我们希望极小化函数

$$F(\mathbf{p}, A) = \sum_{i=1}^{m} \|\mathcal{P}(\mathbf{x}_i - \mathbf{p})\|_2^2. \tag{6.12}$$

一个必要条件是 $\operatorname{grad} F = \mathbf{0}$. 我们首先考虑梯度的第一部分, 即偏导数

$$\frac{\partial F}{\partial \mathbf{p}} = -\sum_{i=1}^{m} 2\mathcal{P}^{\mathrm{T}}\mathcal{P}(\mathbf{x}_i - \mathbf{p}) = -2\mathcal{P}\Big(\sum_{i=1}^{m} \mathbf{x}_i - m\mathbf{p}\Big) = \mathbf{0},$$

其中我们使用了投影算子的性质 $\mathcal{P}^{\mathrm{T}}\mathcal{P} = \mathcal{P}$. 因为 $\mathcal{P}$ 把向量 $\sum_{i=1}^{m} \mathbf{x}_i - m\mathbf{p}$ 投影到 $\mathbf{0}$ 上, 这个向量必须在 A 的值域中, 即

$$\mathbf{p} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i + A\tau. \tag{6.13}$$

把这表达式代入方程 (6.12), 极小化的目标函数简化为

$$G(A) = \sum_{i=1}^{m} \|\mathcal{P}\hat{\mathbf{x}}_i\|_2^2 = \|\mathcal{P}\hat{X}^{\mathrm{T}}\|_F^2, \tag{6.14}$$

我们记

$$\hat{\mathbf{x}}_i = \mathbf{x}_i - \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i,$$

这里我们使用了矩阵的 *Frobenius* 范数 ($\|A\|_F^2 := \sum_{ij} a_{ij}^2$). 因为 $\mathcal{P}$ 是对称的，我们同样能够写为

$$G(A) = \|\hat{X}\mathcal{P}\|_F^2 = \|\hat{X}(I - AA^{\mathrm{T}})\|_F^2 = \|\hat{X} - \hat{X}AA^{\mathrm{T}}\|_F^2. \tag{6.15}$$

如果我们定义 $Y := \hat{X}AA^{\mathrm{T}}$, 它是秩为 s 的矩阵，则我们可以考虑如下的极小化问题

$$\|\hat{X} - Y\|_F^2 = \min \qquad \text{满足} \qquad rank(Y) = s.$$

它是一个著名的问题，即在 *Frobenius* 范数下如何用一个低秩的矩阵逼近当前矩阵 (参见 [2]):

1. 计算 $\hat{X} = U\Sigma V^{\mathrm{T}}$ 的奇异值分解，其中 U, V 是正交的，同时 $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_n)$ 且 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.
2. 则最小化矩阵由 $Y = U\Sigma_s V^{\mathrm{T}}$ 给出，其中

$$\Sigma_s = diag(\sigma_1, \sigma_2, \ldots, \sigma_s, 0, 0, \ldots, 0).$$

现在如果 $Y = U\Sigma_s V^{\mathrm{T}}$, 我们必须找到一个正交矩阵 A 使得 $\hat{X}AA^{\mathrm{T}} = Y$. 容易验证如果我们取 $A = V_1$, 这里 $V_1 = V(:, 1:s)$, 则 $\hat{X}AA^{\mathrm{T}} = U\Sigma_s V^{\mathrm{T}}$. 于是 $\hat{X}$ 的奇异值的分解给了我们所有的低维超平面，它们最佳地拟合了给定点:

$$\mathbf{y} = \mathbf{p} + V_1\mathbf{t}, \quad \text{其中} \quad \mathbf{p} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i.$$

注意到 $V_2 = V(:, s+1:n)$ 同样给了我们超平面的正规形式: 这里超平面被表示为如下线性方程组的解

$$V_2^{\mathrm{T}}\mathbf{y} = V_2^{\mathrm{T}}\mathbf{p}.$$

为了计算超平面，我们需要计算一次奇异值分解. 这可以由 MATLAB 的函数 **hyper.m** 来实现 (算法 6.3). 读者应该注意到语句 `[U S V]=svd(Qt,0)` 以节约的空间计算了奇异值分解. 如果 `Qt` 是一个 $m \times n$ 矩阵 ($m > n$), 则仅仅 U 的前 n 列被计算了，同时 S 是 $n \times n$ 矩阵.

算法 6.3 超平面的计算

```
function [V,p ] =  hyper(Q);
% Fits a hyperplane of dimension s <n
% to a set of given points Q(i,:) belonging
%  to R^n.
% The hyperplane has the equation
% X = p + V(:,1:s)*tau (Parameter Form) or
% is defined as solution of the linear
```

```
% equations  V(:,s+1:n)'*(y - p)=0 (Normal form)
m = max(size(Q));
p = sum(Q)'/m;
Qt = Q - ones(size(Q))*diag(p);
[U S V] = svd(Qt, 0);
```

拟合空间中的平面

假设我们给定空间中的 m 个点 $P_1,\dots,P_m$, 同时我们希望拟合一个通过这些点的平面，使得点到平面距离的平方和最小. 我们能够使用隐式方程描述这个平面

$$c+n_1x+n_2y+n_3z=0,\qquad n_1^2+n_2^2+n_3^2=1. \tag{6.16}$$

在这种情形下，从点 $P=(x,y,z)$ 到平面的距离 r 为

$$r=|c+n_1x+n_2y+n_3z|.$$

从而我们希望极小化

$$\|\mathbf{r}\|=\sum_{i=1}^{m}r_i^2=\min$$

满足

$$\begin{pmatrix} 1 & x_{P_1} & y_{P_1} & z_{P_1} \\ 1 & x_{P_2} & y_{P_2} & z_{P_2} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{P_m} & y_{P_m} & z_{P_m} \end{pmatrix}\begin{pmatrix} c \\ n_1 \\ n_2 \\ n_3 \end{pmatrix}=\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix} \quad 和 \quad n_1^2+n_2^2+n_3^2=1. \tag{6.17}$$

在 MATLAB 中这个问题可以通过调用函数 clsq 容易地被解决:

```
>> [c, n] = clsq ([ones(m,1) P'], 3);
```

另外平面上的一点 P 总能够被明确地表示为

$$P=\mathbf{p}+c_1\mathbf{v}_1+c_2\mathbf{v}_2. \tag{6.18}$$

如果我们使用程序 hyper 计算 $\mathbf{p},\mathbf{v}_1$ 和 $\mathbf{v}_2$, 则 $\mathbf{p}$ 表示这些点的重心，同时 $\mathbf{v}_1$ 和 $\mathbf{v}_2$ 表示平面的两个正交基向量. 附带地，由如下方程给出的直线

$$L=\mathbf{p}+c_1\mathbf{v}_1 \tag{6.19}$$

同样是穿过给定点的最小二乘直线. 我们可以使用 MATLAB 的程序产生一组样本点

```
>> % mainhyper.m
>> m = 100;
>> rand ('seed', 10);
>> randn ('seed', 0);
>> P = rand (2, m) - 0.5;
>> P (3, :) = 0.5 * P (1, :) + 0.5 * P (2, :) + ...
>>           0.25 * randn (1, m);
```

这些点近似地位于平面 $x+y-2z=0$ 上. 在最小二乘意义下穿过这些点的这个平面是通过调用

```
>> [V, p] = hyper (P');
```

来计算的. 我们现在通过

```
>> proj = p * ones (1, m) + ...
>>        V (:, 1:2) * V (:, 1:2)' * (P - p * ones (1, m));
```

来计算点 P_j 到平面上的投影. 为了画出平面的正方形部分, 我们必须知道正方形四个顶点的坐标. 它们能够借助于 MATLAB 如下的语句来计算:

```
>> corners = [1 -1 -1  1
>>            1  1 -1 -1];
>> corners = p * ones (1, 4) + V (:, 1:2) * corners;
```

以同样的方式, 我们能够计算最小二乘直线的两个点如下:

```
>> straight = p * ones (1, 2) + V (:, 1) * [-1 1];
```

现在我们可以把最小二乘直线、最小二乘平面连同所有点在平面上的投影一起绘制出来. 然而所得到的静态图像仅仅给出一个三维状态的不充分的表示. 它很难告诉我们所有对象的相对位置.

另一方面, 一个活动的图像能够给人们更加真实的印象. 在例子中我们可以让平面慢慢地围绕着 z 轴旋转. 这将给我们一个很好的介绍 MATLAB 的 movie 命令的机会.

在 MATLAB 中的动画是由一系列预先算好的画面组成的. 命令 frames= moviein (nframes) 分配一个矩阵 frames, 用它的列来存储动画中所有的画面. 我们用 nframes 表示我们希望产生的所有单个画面的数量. 我们必须事先计算动画的所有画面, 同时调用 getframe 把它们存入 frames. 随后, 我们能够使用命令 movie 显示我们的动画.

为了旋转直线和平面我们引入如下的旋转矩阵

$$R := \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{6.20}$$

其中 $\varphi = 2\pi/n_{\text{frames}}$. 如果我们把矩阵 R 用于点 P, 意味着我们把点 P 绕 z 轴旋转角度 φ.

我们希望使用正交投影来在二维的计算机屏幕上显示三维的图像, 观察者所处的位置在 y 轴的 $-\infty$ 处. 这可以使用 MATLAB 的命令 view(viewmtx (0, 0, 0)) 很容易实现.

为了达到理想的效果, 所有点到平面上的投影, 最小二乘直线和最小二乘平面必须以正确的顺序绘制. 首先, 我们绘制平面后面的投影线. 它们可以由条件 P (2,j) > proj (2, j) 来识别. 其次, 我们通过调用片基元来画平面. 这样的片元通过规定其各个角落的着色来实现表面的明暗处理. 如果我们使用灰度图, 我们可以对平面的表面着色, 使其沿着 y 轴逐渐变深. 这个技术可以实现图形的令人信服的三维效果. 于是, 我们再调用 line 画出最小二乘直线. 最后, 再画出平面前面的投影线. 我们可以通过设置选项 EraseMode 为 none 来避免像元的自动重复绘制, 这在应用中是不希望的. 在这个方式中 MATLAB 没有能力消除缺省隐藏线.

这些考虑导致我们有如下的 MATLAB 程序, 它首先计算所有的画面然后循环演示这个动画:

```
>> nframes = 36;  phi = 2 * pi / nframes;
>> rot = [cos(phi) -sin(phi) 0
>>        sin(phi)  cos(phi) 0
>>        0         0        1];
```

```
>> clf;  axis ('off');  colormap (gray);
>> frames = moviein (nframes);
>> for k = 1:nframes,
>>   clf;
>>   axis ([-1.25 1.25 -1.25 1.25 -1.25 1.25]);  axis ('off');
>>   view (viewmtx (0, 0, 0));  caxis ([-1.3 1.3]);

>>   % plot points behind the plane
>>   for j = 1:m,
>>     if P (2, j) > proj (2, j),
>>       line ([P(1,j) proj(1,j)], ...
>>             [P(2,j) proj(2,j)], ...
>>             [P(3,j) proj(3,j)], ...
>>             'Color', 'yellow', 'EraseMode', 'none');
>>     end
>>   end
>>   drawnow;

>>   % plot plane
>>   patch (corners (1, :), corners (2, :), corners (3, :), ...
>>          -corners (2, :), ...
>>          'EdgeColor', 'none', 'EraseMode', 'none');
>>   drawnow;

>>   % plot least squares line
>>   line (straight (1, :), straight (2, :), straight (3, :), ...
>>         'Color', 'red', 'EraseMode', 'none');
>>      drawnow;

>>   % plot points before the plane
>>   for j = 1:m,
>>     if P (2, j) <= proj (2, j),
>>       line ([P(1,j) proj(1,j)], ...
>>             [P(2,j) proj(2,j)], ...
>>             [P(3,j) proj(3,j)], ...
>>             'Color', 'yellow', 'EraseMode', 'none');
>>     end
>>   end
>>   drawnow;
```

```
>>    frames (:, k) = getframe;

>>    % rotate points
>>    P = rot * P;  proj = rot * proj;
>>    corners = rot * corners;  straight = rot * straight;
>> end;
>> clf;  axis ('off');  movie (frames, inf);
```

参考文献

[1] G. GEISE und S. SCHIPKE, *Ausgleichsgerade, -kreis, -ebene und -kugel im Raum*, Mathematische Nachrichten, 62, 1974.

[2] G.H. GOLUB and CH. VAN LOAN, *Matrix Computations.* 2nd ed., John Hopkins University Press, Baltimore, 1989.

[3] H. SPÄTH, *Orthogonal least squares fitting with linar manifolds.* Numer. Math., 48,1986, pp. 441-445.

[4] VDI BERICHTE 761, *Dimentional Metrology in Production and Quality Control.* VDI Verlag, Düsseldorf, 1989.

[5] H.J.WARNECKE und W.DUTSCHKE, *Fertigungsmesstechnik*, Springer, Berlin, 1984.

第七章　广义台球问题

S. Bartoň

7.1　引言

我们考虑如下的问题：给定一个台球桌 (不一定是长方形的) 和桌上的两个球，问向哪个方向击第一个球使得它从台球桌的边缘弹回并且撞到第二个球？这个问题较早在圆形球桌的情形下在 [1,3] 中被解决了.

令台球桌的形状被 $X = f(t), Y = g(t)$ 表示为参数形式. 这些函数可以是任意的，只要求它们存在一阶导数. 我们的目的在于求出由坐标 $[X(t_i), Y(t_i)]$ 表示的边界上的点，我们将使用两个不同的方法求解此问题：广义反射法和最短轨线法. 我们使用两个方法得出参数 t 的函数，它的根是点 $t_1, \ldots, t_n$, 即给定问题的解. 在本章的第一部分我们将首先求出解析解. 在第二部分我们将通过求解某些实际的例子来直观地检验这个解. 在更复杂的情形下，我们将使用数值方法来求解最终的方程.

7.2　广义反射法

问题的解可以分两步得到. 首先，求出由球桌边缘一点反射的球的轨迹. 其次，求出球桌边界的点使得第二个球刚好在这条轨迹上.

第一步可以通过推广平面反射问题来解决，即按照反射定律求从点 P 沿着直线 l 到点 Q 的路径. 这里的反射定律是指投射角等于反射角.

在第二步中，我们计算反射的轨迹与第二个球位置的距离. 这个距离是撞击位置点的函数. 一旦相应的距离等于零的点被求出，问题也就解决了.

7.2.1　直线和曲线反射

首先，我们求出点 P 的镜像点 M, 对称轴为直线 l. 连接点 M 和 Q 的直线 l' 与直线 l 相交于点 T. 现在，所要求的通过点 T 的路径如图 7.1 所示. 为了击中位于点 Q 的球必须从点 P 瞄准点 T. 我们使用球台边界的切线作为镜面线. 我们可以用隐式的方式把它表示为

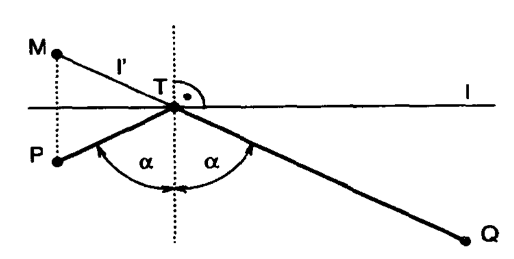
$$l \equiv l_1 \equiv y - T_y = k(x - T_x), \qquad k_1 \equiv k = \left. \frac{\frac{\mathrm{d}X(t)}{\mathrm{d}t}}{\frac{\mathrm{d}Y(t)}{\mathrm{d}t}} \right|_T \tag{7.1}$$

其中

$$T \equiv [T_x, T_y] = [X(t), Y(t)],$$

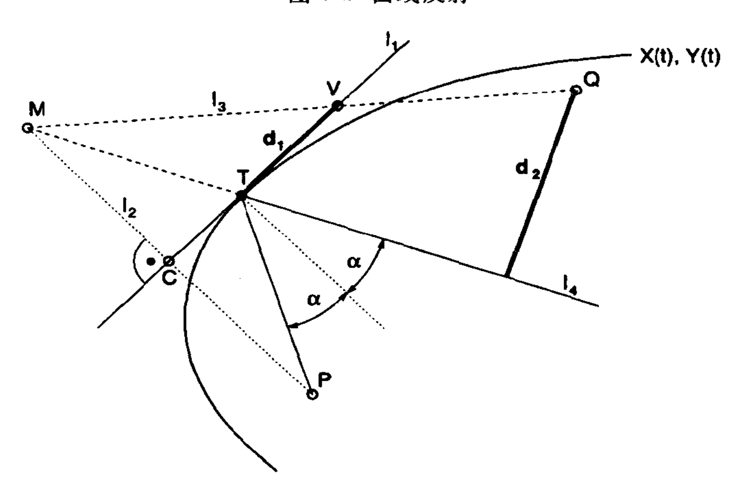
T_x, T_y 是台球桌边缘上点 T 的坐标， k 是边界函数在这个点的导数. 击到边界的这个球刚好就像由这条线反射一样被反弹回去. 我们使用 l 作为镜面线去求一个球的位置的镜像. 图 7.2 给出了广义台球问题的解. 我们能够把位于点 P 的球向边界的点 T 瞄准. 我们使用在点 T 处边界的切线 l_1 为镜面线求出点 P 的镜像 M. l_2 是连接点 P 和 M 的线段. l_1 垂直于 l_2, 与 l_2 相交于它的中点 C.

图 7.1 直线反射



如果点 M 已知，则可以画一条连接点 M 和 Q 的线段 l_3，和射线 l_4(从点 M 出发穿过点 T)．位于点 P 的球在反射之后将沿着这条射线运动．现在我们可以计算点 Q 到射线 l_4 的距离 d_2．如果这个距离等于零，即 $d_2=0$，则第二个球的位置刚好在第一个球反射以后的轨道上．因此这两个球一定相撞．第二个可能性是计算点 T 和直线 l_1 与 l_3 的交点 V 之间的距离 d_1．要想击到位于点 Q 的球，这个距离必须要等于零，即 $d_1=0$，因为在这个情形下 T 和 V 一定重合，l_4 也将与 l_3 重合．计算 d_1 要比计算 d_2 简单．确定两点之间的距离要比计算点与直线的距离简单．于是问题就简化为求点 T 使得对应的距离 d_1 或 d_2 分别等于零．

图 7.2 曲线反射



7.2.2 数学描述

为了得到解析解我们再一次考虑图 7.2．我们使用同一类型的公式来描述所有的直线，这样对简化分析过程是有益的．直线可以用它的点斜式来表示．对于第一条直线 l_1 我们用方程 (7.1) 来表示．其它的直线被表示为

$$\begin{aligned} l_2 &\equiv y-P_y=k_2(x-P_x), \quad \text{其中} \quad k_2 = -\frac{1}{k_1} \\ l_3 &\equiv y-Q_y=k_3(x-Q_x), \quad \text{其中} \quad k_3 = \frac{M_y-Q_y}{M_x-Q_x} \end{aligned} \tag{7.2}$$

求得点 M 坐标的最简便方法是找到直线 l_1 和 l_2 的交点 C, 同时使用镜面对称的条件

$$C_x = \frac{P_x + M_x}{2}, \quad C_y = \frac{P_y + M_y}{2}. \tag{7.3}$$

随后就有可能计算直线 l_1 和 l_3 的交点 V 的坐标，同时确定瞄准误差距离 d_1

$$d_1 = \sqrt{(V_x - T_x)^2 + (V_y - T_y)^2}. \tag{7.4}$$

使用方程 (7.1) 和 (7.4) 我们能够计算出 d_1 作为台球位置坐标、球台形状函数及其导数的函数

$$d_1(t) = d_1\big(P_x, P_y, Q_x, Q_y, X(t), Y(t), \frac{\mathrm{d}X(t)}{\mathrm{d}t}, \frac{\mathrm{d}Y(t)}{\mathrm{d}t}\big). \tag{7.5}$$

我们希望关于 t 求解方程 $d_1(t) = 0$.MAPLE 可以用来给出 $d_1(t)$ 明确的表达式同时求出它的解.

表 7.1 广义台球问题中使用的主要变量

变量	意义
T_x, T_y	$T_x = X(t), T_y = Y(t)$, 球台形状的描述
k_1	球台函数在点 $[T_x, T_y]$ 切线的斜率 (直线 l_1)
k_2	直线 l_2 的斜率，它垂直于直线 l_1, $[P_x, P_y]$ 是 l_2 上的点
k_3	连接点 $[Q_x, Q_y]$ 和 $[M_x, M_y]$ 的直线 l_3 的斜率
$Aimerr$	点 $[V_x, V_y]$ 和 $[T_x, T_y]$ 之间距离的平方 (瞄准误差)
DDA, DB	$Aimerr$ 的分子和分母
P_x, P_y	第一个球的坐标
Q_x, Q_y	第二个球的坐标
M_x, M_y	点 $[P_x, P_y]$ 的镜像点，镜线是 l_1
C_x, C_y	直线 l_1 和 l_2 交点的坐标
V_x, V_y	直线 l_1 和 l_3 交点的坐标

7.2.3 MAPLE 解

程序是严格地以上面提到的关系式为基础的. 我们将使用表 7.1 所列的变量. 广义台球问题在 $[V_x, V_y] \equiv [T_x, T_y]$ 时，即 $DDA = 0$ 就被解出.

```
> E1 := solve({V[y] - Q[y] = k[3]*(V[x]-Q[x]),
>       V[y] - T[y] = k[1]*(V[x] - T[x])}, {V[x], V[y]}):
> assign(E1):
> E2 := solve({C[y] - P[y] = k[2]*(C[x] - P[x]),
>       C[y] - T[y] = k[1]*(C[x] - T[x])}, {C[x], C[y]}):
> assign(E2):
> M[x] := P[x] + 2*(C[x] - P[x]):  M[y] := P[y] + 2*(C[y] - P[y]):
> k[3] := (M[y] - Q[y])/(M[x] - Q[x]):
> T[y] := Y(t):  T[x] := X(t):
> k[2] := -1/k[1]:
> k[1] := diff(Y(t), t)/diff(X(t),t):
> Aimerr := normal((T[x] - V[x])^2 + (T[y] - V[y])^2):
```

我们现在需要求解方程 $Aimerr = 0$. 如果我们使用球台边界的参数表达式 $X(t)$ 和 $Y(t)$, $Aimerr$ 的表达式就变成一个分数. 它能够被用来简化问题的求解，变成仅仅需要去求分子等于零的解. 但是需要检验这样一个零点是不是正确的解，我们必须确认 $Aimerr$ 的分子和分母没有共同的零点.

现在我们尝试简化 *Aimerr* 的分子.

```
> DA := op(1, numer(Aimerr)):
> DB := denom(Aimerr):
> DA1 := factor(select(has, DA, {diff(X(t), t)^2,
>        diff(Y(t), t)^2})):
> Da2 := select(has, DA, diff(X(t), t)):
> DA2 := factor(select(has, Da2, diff(Y(t), t))):
> DA - expand(DA2 + DA1);
```

$$0$$

```
> DAA := DA1 + DA2;
```

$$\begin{aligned}DAA := &-2(\frac{\partial}{\partial t}\mathrm{Y}(t))(\frac{\partial}{\partial t}\mathrm{X}(t))\\&(Q_y\,\mathrm{Y}(t)-Q_y\,P_y-\mathrm{Y}(t)^2+\mathrm{X}(t)^2+P_x\,Q_x-\mathrm{X}(t)\,Q_x+\mathrm{Y}(t)\,P_y-P_x\,\mathrm{X}(t))\\&+((\frac{\partial}{\partial t}\mathrm{Y}(t))-(\frac{\partial}{\partial t}\mathrm{X}(t)))((\frac{\partial}{\partial t}\mathrm{Y}(t))+(\frac{\partial}{\partial t}\mathrm{X}(t)))\\&(-Q_y\,P_x+Q_y\,\mathrm{X}(t)-P_y\,Q_x+\mathrm{Y}(t)\,P_x-2\,\mathrm{Y}(t)\,\mathrm{X}(t)+\mathrm{Y}(t)\,Q_x+\mathrm{X}(t)\,P_y)\end{aligned}$$

MAPLE 代码的结果是变量 *DDA*, 它依赖于球台形状使用的特定的函数. 对于任何所提的实例问题, 这些函数必须以输入的形式提供.

7.3 最短轨线法

从物理学可知 [2] 从一个点发出到另一个点的光线的轨道将极小化路径的长度. 在反射的情形下这条曲线由两段直线组成. 第一段直线连接光源点 (在我们的情形下: 第一个球的位置) 和反射点 (球台边界上的点). 后一个点由第二条线连向观测点 (第二个球的位置). 使用这个思想, 我们同样能够解决广义台球问题.

我们可以给出连接点 P 和球台边界上的反射点 T 的第一条线段 l_P, 和连接点 Q 和 T 的第二条线段 l_Q. 现在我们必须求出这两条线段的最小和 S, 它是 T 的函数. 请记住 T 是球台边界上的点. 注意到 $T\equiv[X(t),Y(t)]$, 我们的问题是关于参数 t 求解方程

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\sqrt{(T_x(t)-P_x)^2+(T_y(t)-P_y)^2}+\sqrt{(T_x(t)-Q_x)^2+(T_y(t)-Q_y)^2}\right)=0$$

7.3.1 MAPLE 解

```
> LP := sqrt((X(t) - P[x])^2 + (Y(t) - P[y])^2):
> LQ := sqrt((X(t) - Q[x])^2 + (Y(t) - Q[y])^2):
> lP := diff(LP,t):  lQ := diff(LQ,t):
> dll := numer(lP)*denom(lQ) + numer(lQ)*denom(lP);
```

$$\begin{aligned}dll := &(-(\frac{\partial}{\partial t}\mathrm{X}(t))\,P_x+(\frac{\partial}{\partial t}\mathrm{X}(t))\,\mathrm{X}(t)+\mathrm{Y}(t)\,(\frac{\partial}{\partial t}\mathrm{Y}(t))-(\frac{\partial}{\partial t}\mathrm{Y}(t))\,P_y)\\&\sqrt{(\mathrm{X}(t)-Q_x)^2+(\mathrm{Y}(t)-Q_y)^2}+\\&((\frac{\partial}{\partial t}\mathrm{X}(t))\,\mathrm{X}(t)-(\frac{\partial}{\partial t}\mathrm{X}(t))\,Q_x+\mathrm{Y}(t)\,(\frac{\partial}{\partial t}\mathrm{Y}(t))-Q_y\,(\frac{\partial}{\partial t}\mathrm{Y}(t)))\sqrt{(\mathrm{X}(t)-P_x)^2+(\mathrm{Y}(t)-P_y)^2}\end{aligned}$$

为了求解最后的方程 $dll=0$, 我们必须首先确定球台形状的函数 $X(t)$ 和 $Y(t)$.

7.4 例子

我们现在把反射法与最短轨线法相比较. 两者都得到同样的结果. 作为第一个例子, 我们考虑圆形的球台, 它的解是大家熟知的 [1,3].

7.4.1 圆形球台

为检验我们的程序, 我们将再一次在同样的假设, 即 $R=1, Q_y=0$ 下计算 [1,3] 中给出的解. 我们能够通过定义

```
> X(t) := cos(t):  Y(t) := sin(t):  Q[y]:= 0:
```

继续在 7.2.3 和 7.3.1 中所提出的 MAPLE 程序的计算. 因为球台的形状是确定的, 球的位置完全任意. 首先我们将求解 $DAA=0$(参见 7.2.3).

```
> DA := simplify(DAA);
```

$$DA := 2\sin(t)\cos(t)\,P_x\,Q_x + \cos(t)\,P_y - 2\cos(t)^2\,P_y\,Q_x + P_y\,Q_x - \sin(t)\,P_x - \sin(t)\,Q_x$$

```
> SolDA := solve(DA,t);
```

$$SolDA := \arctan(\frac{P_y\,(-\%1 + 2\,\%1^2\,Q_x - Q_x)}{2\,\%1\,P_x\,Q_x - P_x - Q_x}, \%1)$$
$$\%1 := \mathrm{RootOf}((4\,P_y{}^2\,Q_x{}^2 + 4\,P_x{}^2\,Q_x{}^2)\,_Z^4 + (-4\,P_y{}^2\,Q_x - 4\,P_x{}^2\,Q_x - 4\,P_x\,Q_x{}^2)\,_Z^3$$
$$+ (2\,P_x\,Q_x - 4\,P_x{}^2\,Q_x{}^2 + P_x{}^2 + Q_x{}^2 + P_y{}^2 - 4\,P_y{}^2\,Q_x{}^2)\,_Z^2$$
$$+ (4\,P_x{}^2\,Q_x + 4\,P_x\,Q_x{}^2 + 2\,P_y{}^2\,Q_x)\,_Z - Q_x{}^2 + P_y{}^2\,Q_x{}^2 - P_x{}^2 - 2\,P_x\,Q_x)$$

使用命令 solve 我们能够得出一个参数 t 的解析值使得 $DA=0$. 为简洁起见, 这里没有提供求解的细节. 为比较我们的结果和 [1,3] 中得到的结果, 我们将代替一般的变量 P_x, P_y, Q_x 为它们的数值. 我们将检验 DA 的零点是不是 $Aimerr$ 的正确解. 如果它们代入 DB, 值不为零, 解就是正确的.

```
> P[x] := 1/2:  P[y] := 1/2:  Q[x] := -3/5:
> DA := simplify(DA):
> SolDA := solve(DA, t);
```

$$SolDA := \arctan(-2\,\%1 - 1 + 6\,\%1^3 + 5\,\%1^2, \%1)$$
$$\%1 := \mathrm{RootOf}(-23\,_Z^2 + 24\,_Z^3 - 9\,_Z + 36\,_Z^4 + 4)$$

```
> FSolDA := evalf(allvalues(SolDA));
```

$$FSolDA := .9380770491, 2.750972642, -1.251457483, 2.274796770$$

```
> for i from 1 by 1 to nops([FSolDA]) do:
>     tA[i] := FSolDA[i];
>     DBi := evalf(subs(t = tA[i], DB));
>     if DBi = 0 then print(`Root`,i,`unusable !`); fi;
>     print(`i =`,i, ` tA = `, tA[i],` ==> DB = `, DBi);
> od:
```

$$i\ =,1,\ \ tA\ =,.9380770491,\ \ ==>\ DB\ =,2.742080139$$
$$i\ =,2,\ \ tA\ =,2.750972642,\ \ ==>\ DB\ =,2.948610256$$
$$i\ =,3,\ \ tA\ =,2.274796770,\ \ ==>\ DB\ =,2.415361451$$
$$i\ =,4,\ \ tA\ =,-1.251457483,\ \ ==>\ DB\ =,6.280614817$$

注意到目标点的位置角与 [1,3] 中相同, 即 `evalf(tA[4]+2*Pi)=5.031727826`. 对于 $t = t_{A1},\ldots,t_{A4}$ 的四个目标点的坐标可以通过 $T_x = \cos(t), T_y = \sin(t)$ 计算出来.

我们现在检验第二个方法 $dl = 0$(参见 7.3.1). 我们再次使用 MAPLE 去求一般的解析解:

```
> P[x] := 'P[x]':  P[y] := 'P[y]':  Q[x] := 'Q[x]':  Q[y] := 0:
> dl := simplify(dll);
```

$$dl := \sqrt{-2\cos(t)\,Q_x + {Q_x}^2 + 1}\,\sin(t)\,P_x - \sqrt{-2\cos(t)\,Q_x + {Q_x}^2 + 1}\,\cos(t)\,P_y + \sin(t)\,Q_x\,\sqrt{-2\cos(t)\,P_x + {P_x}^2 + 1 - 2\sin(t)\,P_y + {P_y}^2}$$

```
> # Soldl := solve(dl,t):
```

解析解是存在的, 但是它要比前面的例子复杂, 因为变量 dl 包含有平方根. 在这个情况下最好首先来变更这个方程, 即在求解之前把平方根去掉.

```
> eq := op(1, dl) + op(2, dl) = -op(3, dl):
> eq := expand(map(u -> u^2, eq)):
> eq := lhs(eq) - rhs(eq) = 0:
> Soldl := solve(eq, t):
```

这个结果并不特别复杂, 但为了简单起见, 这里也没有把它列出来. 我们可以再次分配 P_x, P_y, Q_x 的值同时再次求解 eq.

```
> P[x] := 1/2:  P[y] := 1/2:  Q[x] := -3/5:
> eq;
```

$$\frac{33}{50}\sin(t)^2\cos(t) - \frac{1}{5}\sin(t)^2 - \frac{3}{5}\sin(t)\cos(t)^2 - \frac{17}{25}\sin(t)\cos(t) + \frac{3}{10}\cos(t)^3 + \frac{17}{50}\cos(t)^2 + \frac{9}{25}\sin(t)^3 = 0$$

```
> Soldl := solve(eq,t);
```

$$Soldl := \arctan\left(\frac{\frac{33}{146} + \frac{5}{146}\sqrt{137}}{-\frac{15}{146} + \frac{11}{146}\sqrt{137}}\right),\ \arctan\left(\frac{\frac{33}{146} - \frac{5}{146}\sqrt{137}}{-\frac{15}{146} - \frac{11}{146}\sqrt{137}}\right) - \pi,$$

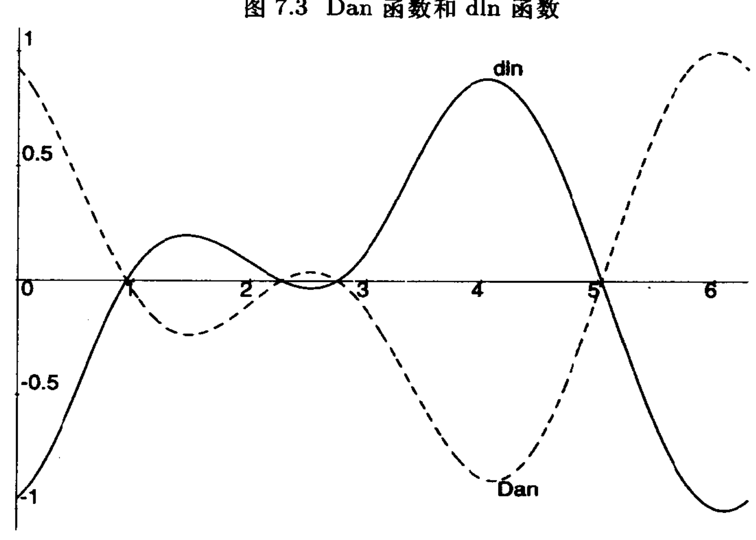
$$\arctan(-2\,\%1 - 1 + 6\,\%1^3 + 5\,\%1^2, \%1)$$
$$\%1 := \mathrm{RootOf}(-23\,_Z^2 + 24\,_Z^3 - 9\,_Z + 36\,_Z^4 + 4)$$

```
> FSoldl := evalf(allvalues({Soldl}));
```

$$FSoldl := \{.6775335699, .9380770491, -2.965871238\},$$
$$\{.6775335699, 2.750972642, -2.965871238\},$$
$$\{.6775335699, -2.965871238, -1.251457483\},$$
$$\{.6775335699, -2.965871238, 2.274796770\}$$

由此可见两个解是相同的. 注意到第二个解的多余的根是由于方程的变形产生的, 它对于我们没有实用价值.

图 7.3 Dan 函数和 dln 函数



我们比较函数 $DA(t)$ 和 $dl(t)$. 因为它们被做了很大的化简, 已经失去了原始的物理意义. 我们首先将它们除以它们最大的绝对值以把这些函数正规化. 这样我们就可以把这两个函数绘制在同一个坐标上, 如图 7.3 所示.

```
> DAd := diff(DA, t):  dld := diff(dl, t):
> tamax := fsolve(DAd = 0, t, 6..7):
> tlmax := fsolve(dld = 0, t, 6..7):
> Dan := DA/abs(subs(t = tamax, DA)):
> dln := dl/abs(subs(t = tlmax, dl)):
> with(plots):
> p1 := plot(Dan, t = 0..2*Pi, linestyle=3):
> p2 := plot(dln, t = 0..2*Pi, linestyle=0):
> display({p1,p2});
```

粗略看去, 似乎两个函数 $Dan(t)$ 和 $dln(t)$ 实质上是相同的, 只是符号不同. 我们通过绘制一条不寻常的曲线 $PF(x,y) \equiv [Dan(t), dln(t)]$ 来发现它们的区别 (参见图 7.4). 如果我们的猜想是正确

的，这个图形应该是从点 [-1,1] 到点 [1,-1] 的直线段．然而所观察到的与这条线段的偏差表明这两个函数事实上是不同的.

```
> plot([Dan, dln, t = 0..2*Pi]);
```

有趣的是得到了如下的新的曲线:

$$\tilde{X}(t) = X(t)(1 + Dan), \quad \tilde{Y}(t) = Y(t)(1 + Dan). \tag{7.6}$$

这条曲线是由球台形状函数加上函数 Dan(或 dln) 作为“干扰”构成的. 因为在目标点函数 DA, Dan 等于零, 这些点的坐标是在边界上 (图 7.5). 因此目标点能够作为球台形状函数和干扰形状函数 (7.6) 的交点从图像上求出.

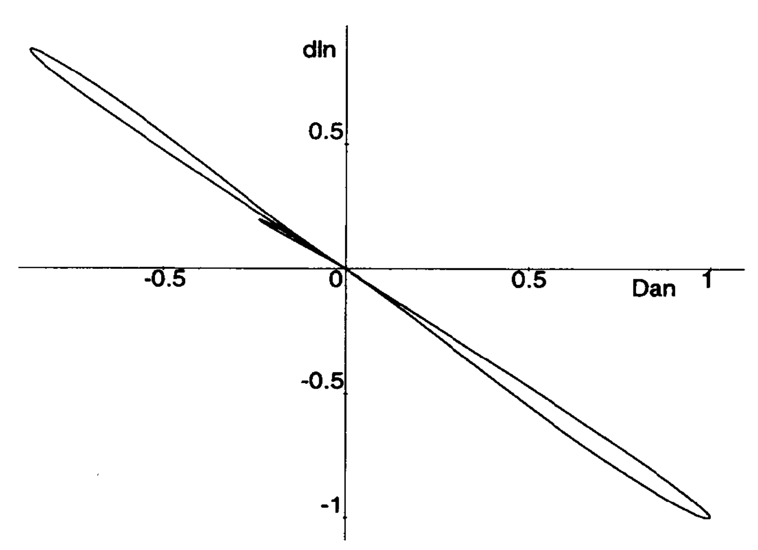
算法 7.1 **ResultPlot** *过程*

```
ResultPlot := proc(P1x, P1y, P2x, P2y, Xborder, Yborder, Perturb,
                   Start, End, Zerosvec, Nzeros, Xrange, Yrange)
  local i, n, Trajectories, AimPointX, AimPointY,
            PlotTraject, PlotPoints, PlotBorder, PlotPert;
  for i from 1 by 1 to Nzeros do
     AimPointX[i] := evalf(subs(t = Zerosvec[i], Xborder));
     AimPointY[i] := evalf(subs(t = Zerosvec[i], Yborder));
     Trajectories[i] := [[P1x, P1y], [AimPointX[i],
                         AimPointY[i]], [P2x, P2y]];
  od;
  PlotTraject := plot({seq(Trajectories[i], i = 1..Nzeros)},
                      style = LINE, thickness=1):
  PlotPoints := plot({[P1x, P1y], [P2x, P2y],
                      seq([AimPointX[i], AimPointY[i]],
                       i = 1..Nzeros)}, style=POINT):
  PlotBorder := plot([Xborder, Yborder, t = Start..End],
                 thickness=2):
  PlotPert := plot([Xborder*(1 + Perturb), Yborder*(1 + Perturb),
                    t = Start..End], linestyle=3):
  display({PlotTraject, PlotPoints, PlotBorder, PlotPert},
        scaling=constrained, view=[Xrange, Yrange]);
end;
```

因为图像输出多次被使用，我们将产生一个 MAPLE 程序 ResultPlot(算法 7.1)，它将被用来描绘这个图像解．程序 ResultPlot 很容易使用．例如图 7.5 就是用它绘出的.

```
> ResultPlot(P[x], P[y], Q[x], Q[y], X(t), Y(t),
>            Dan, 0, 2*Pi, tA, 4, -1..2, -1.25..1);
```

图 7.4 Dan 和 dln 的图像比较



7.4.2 椭圆球台

具有半轴 a 和 b 的一个椭圆可以由如下的参数方程描述:

$$X(t) = a\cos(t), \quad Y(t) = b\sin(t). \tag{7.7}$$

我们考虑长半轴为 $a=1$, 短半轴为 $b=0.8$ 的椭圆. 令 $P \equiv [-0.6, -0.3]$ 和 $Q \equiv [0.5, 0.5]$ 是球的位置的坐标. 仍然使用前面得到的方程 DA(参见 7.2.3) 和 dl(参见 7.3.1).

```
> X(t) := cos(t):  Y(t) := 4/5*sin(t):
> P[x] := -3/5:  P[y] := -3/10:  Q[x] := 1/2:  Q[y] := 1/2:
> DA := simplify(DAA);
```

$$DA := \frac{42}{125}\sin(t)\cos(t) + \frac{18}{625}\sin(t)\cos(t)^2 + \frac{369}{500}\cos(t)^2 + \frac{9}{125}\cos(t)^3 + \frac{2}{25}\sin(t) + \frac{7}{125}\cos(t) - \frac{9}{20}$$

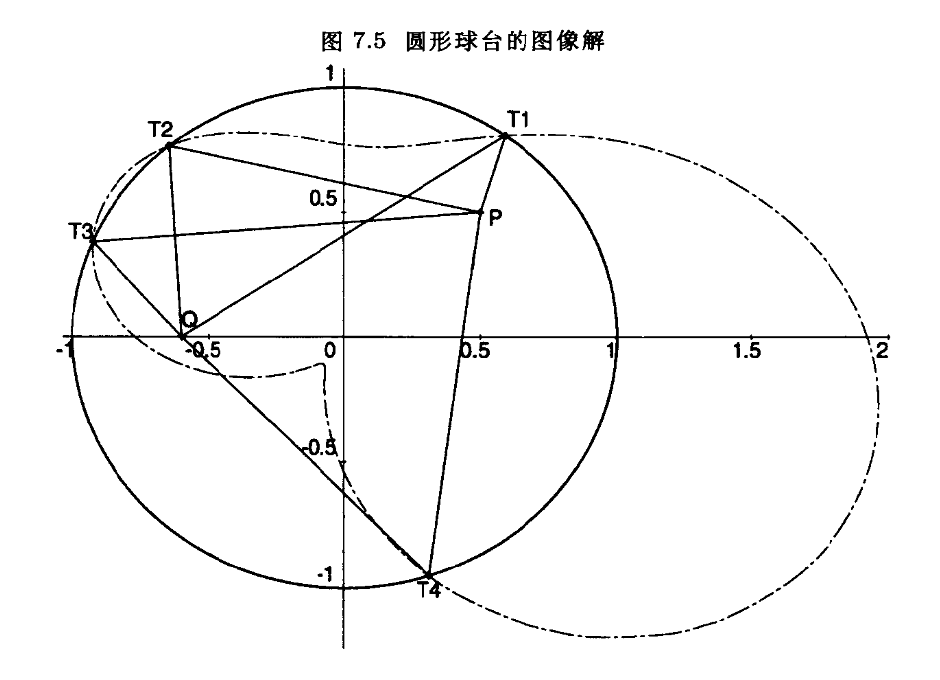
```
> SolDA := solve(DA,t);
```

$$SolDA := \arctan(\frac{60534443}{18363160}\%1 + \frac{32992953}{45907900}\%1^2 - \frac{586629}{1836316} - \frac{2457054}{57384875}\%1^5 - \frac{10344267}{11476975}\%1^4 - \frac{2162333043}{459079000}\%1^3, \%1)$$
$$\%1 := \mathrm{RootOf}(-4826050_Z^2 + 326640_Z^3 + 4183641_Z^4 - 651000_Z + 785160_Z^5 + 37584_Z^6 + 1225625)$$

```
> irreduc(op(1,op(2,SolDA)));
```

$$true$$

图 7.5 圆形球台的图像解



可以看到，这个问题得不出解析解. 于是我们使用数值方法求解.

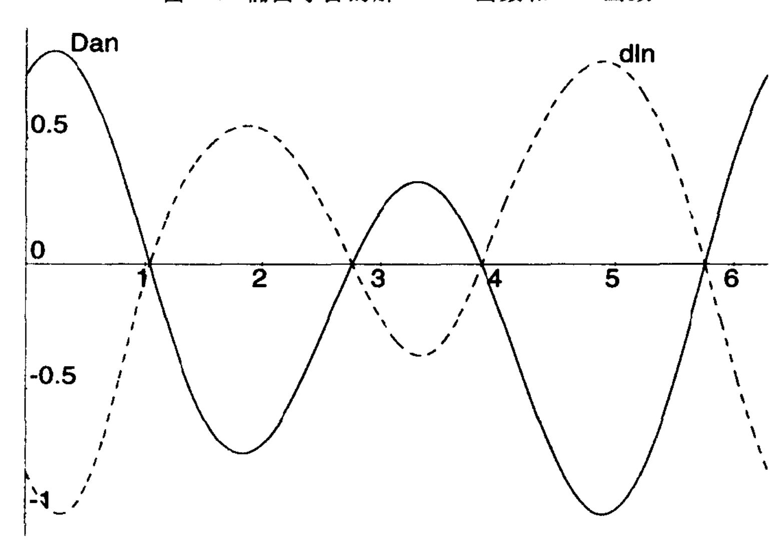
```
> tA[1] := fsolve(DA, t, -3..-2);
> tA[2] := fsolve(DA, t, -1..0);
> tA[3] := fsolve(DA, t, 1..2);
> tA[4] := fsolve(DA, t, 2..3);
> tl[1] := fsolve(dll, t, -3..-2);
> tl[2] := fsolve(dll, t, -1..0);
> tl[3] := fsolve(dll, t, 1..2);
> tl[4] := fsolve(dll, t, 2..3);
```

$$tA_1 := -2.425591133$$
$$tA_2 := -.5260896824$$
$$tA_3 := 1.038696884$$
$$tA_4 := 2.761693514$$
$$tl_1 := -2.425591133$$
$$tl_2 := -.5260896824$$
$$tl_3 := 1.038696884$$
$$tl_4 := 2.761693514$$

```
> Dan := DA/abs(evalf(subs( t = fsolve(diff(DA, t),
>               t, -2..-1), DA))):
> dln := dll/abs(evalf(subs( t = fsolve(diff(dll, t),
>                t, -2..-1), dll))):
> p1:=plot(Dan, t = 0..2*Pi,linestyle=0):
> p2:=plot(dln, t = 0..2*Pi,linestyle=3):
> display({p1,p2});
> ResultPlot(P[x], P[y], Q[x], Q[y], X(t), Y(t),
```

```
>             Dan, -Pi, Pi, tA, 4, -1.5..2, -0.8..0.8);
```

图 7.6 椭圆球台的解 –Dan 函数和 dln 函数



7.4.3 蜗线形球台

蜗线或螺线形的球台是很少见的，但是它能被用来展示 MAPLE 的能力．一个“蜗形曲线”或 Archimedes 螺线最好是在极坐标系下用方程 $\varrho = a\varphi$ 来描述．转换为直角坐标系，我们可以给出 Archimedes 螺线的参数表示： $X(t) = t\cos(t), Y(t) = t\sin(t)$, 其中 $a = 1$. 这些方程可以用于求解蜗线球台问题．令两个球的坐标为 $P \equiv [-6, -12], Q \equiv [7, 5]$.

```
> X(t) := t*cos(t):  Y(t) := t*sin(t):
> P[x] := -6:  P[y] := -12:  Q[x] := 7:  Q[y] := 5:
> DA := simplify(DAA):
> Dan := DA/abs(evalf(subs(t = fsolve(diff(DA, t),
>                t, 15..17), DA))):
> dln := dll/abs(evalf(subs(t = fsolve(diff(dll, t),
>                 t, 15..17), dll))):
> p1:=plot(Dan, t = 4*Pi..6*Pi,linestyle=0):
> p2:=plot(dln, t = 4*Pi..6*Pi,linestyle=3):
> display({p1,p2});
> p1:=plot(Dan, t = 12.5..12.85,linestyle=0):
> p2:=plot(dln, t = 12.5..12.85,linestyle=3):
> display({p1,p2});
> tA[1] := fsolve(DA, t, 12.5..12.6);
> tA[2] := fsolve(DA, t, 12.7..12.8);
> tA[3] := fsolve(DA, t, 14..15);
> tA[4] := fsolve(DA, t, 16..17);
```

$$tA_1 := 12.56987562$$
$$tA_2 := 12.75992609$$

图 7.7 椭圆球台的图像解

$$tA_3 := 14.95803948$$
$$tA_4 := 16.82749442$$

```
> ResultPlot(P[x], P[y], Q[x], Q[y], X(t), Y(t),
>            dln, 4*Pi, 6*Pi, tA, 4, -20..30, -31..26);
```

结果以与前面相同的方式提供. 图 7.8 显示函数 *Dan* 和 *dln* 并带有放大的细节. 注意到这里有两个很接近的函数零点. 这图像有助于寻求 fsolve 的初始值.

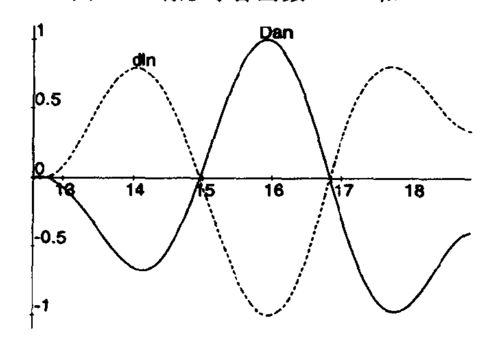


图 7.8 蜗形球台函数 *Dan* 和 *dln*

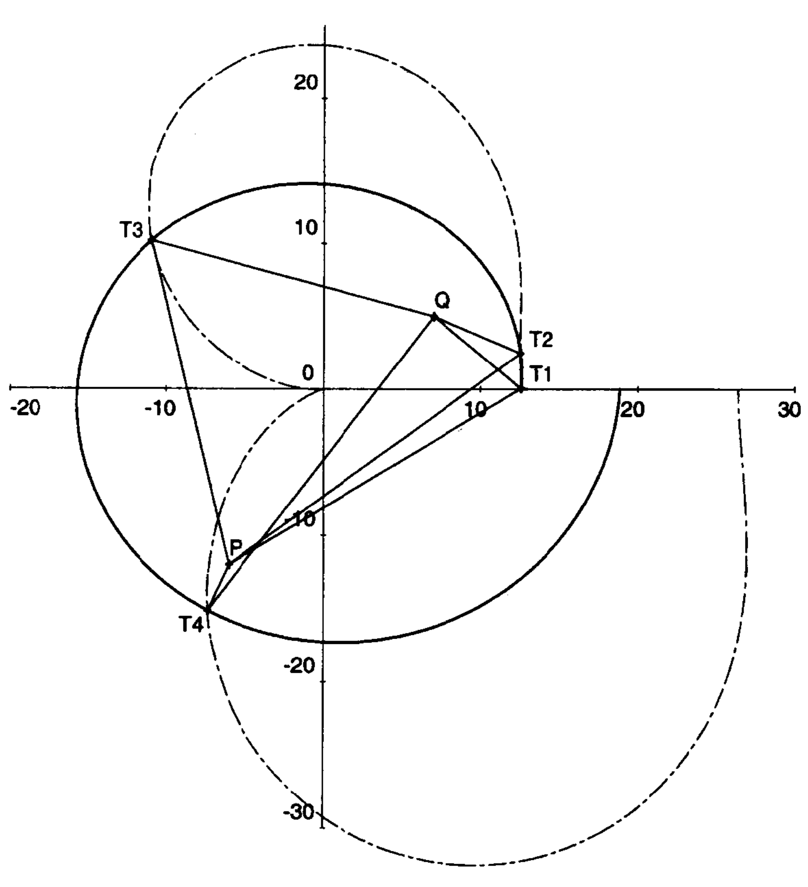
图 7.9 *Dan* 和 *dln* 的放大细节

7.4.4 星形球台

最后我们考虑一个有些古怪的球台形状, 一个五点星形的球台 (非常适合于肥胖人使用的球台). 这个形状能够被如下的参数方程成功地近似:

$$X(t) = \cos(t)\big(1 + \frac{\sin(5t)}{5}\big), \qquad Y(t) = \sin(t)\big(1 + \frac{\sin(5t)}{5}\big).$$

图 7.10 蜗形球台的图像解



令球的坐标是 $P \equiv [-0.8, 0.2]$ 和 $Q \equiv [0.6, -0.8]$. 这样这些球的位置接近星形角的顶端 (图 7.13). 我们将象前面一样求解这个例子. 但是我们将看到, 函数 DA 和 dll 将有许多零点.

```
> X(t) := cos(t)*(1 + sin(5*t)/5):
> Y(t) := sin(t)*(1 + sin(5*t)/5):
> P[x] := -4/5:  P[y] := 1/5:  Q[x] := 3/5:  Q[y] := -4/5:
> plot({DAA, dll}, t = 0..2*Pi);
> Dan := DAA/evalf(abs(subs(t = fsolve(diff(DAA, t),
>            t, 1..1.5), DAA))):
> dln := dll/evalf(abs(subs(t = fsolve(diff(dll, t),
>            t, 1.7..1.9), dll))):
> p1:=plot(Dan,t = 0..2*Pi,linestyle=0):
> p2:=plot(dln,t = 0..2*Pi,linestyle=3):
> display({p1,p2});
> plot([Dan, dln, t = 0..2*Pi]);
```

图 7.11 和图 7.12 用来显示描述目标点位置的两个不同的函数 DAA 和 dll. 如图所示, 这里一共有十个目标点, 但其中有些是不能被接受的, 因为台球的轨道与球台的边界交叉. 我们将通过检查

图形解来求出正确的目标点.

图 7.11. 星形球台的函数 Dan 和 dln

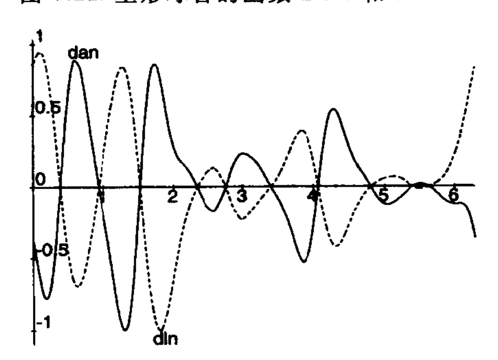


图 7.12. Dan 和 dln 的参数图

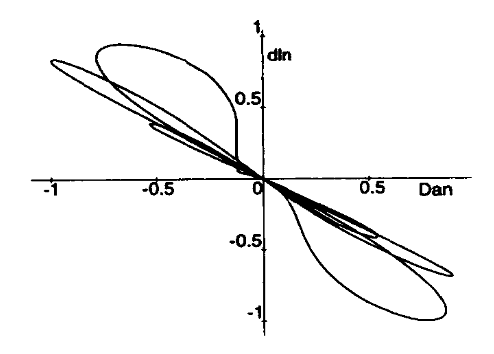
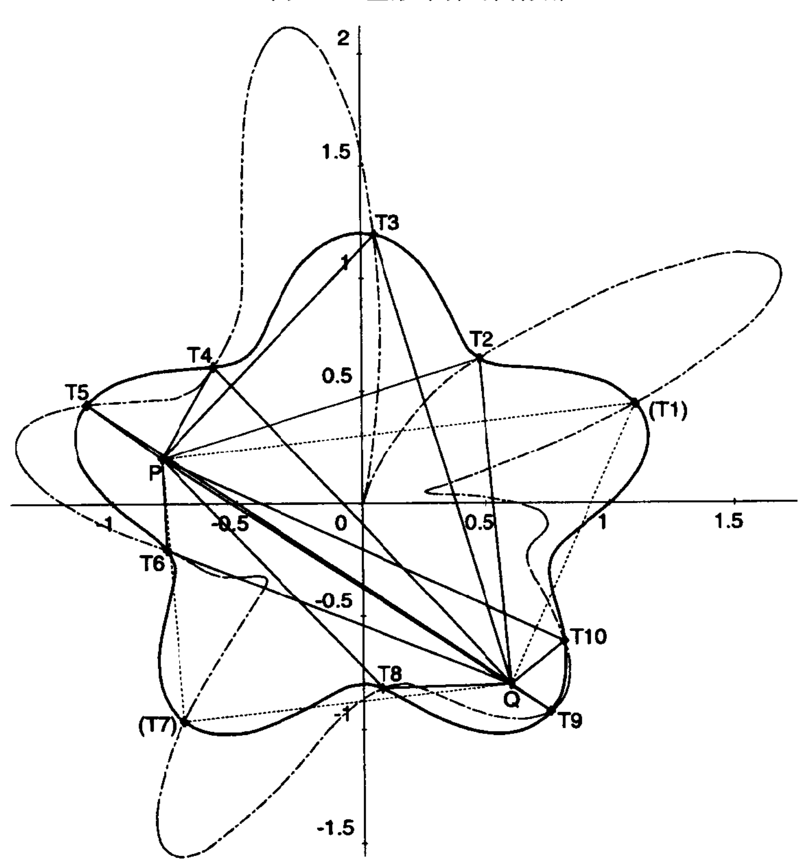


图 7.13 星形球台的图像解




```
> tA[1] := fsolve(DAA = 0, t, 0.3..0.4);
> tA[2] := fsolve(DAA = 0, t, 0.9..1.0);
> tA[3] := fsolve(DAA = 0, t, 1.5..1.6);
> tA[4] := fsolve(DAA = 0, t, 2.3..2.4);
> tA[5] := fsolve(DAA = 0, t, 2.7..2.8);
```

```
> tA[6] := fsolve(DAA = 0, t, 3.3..3.5);
> tA[7] := fsolve(DAA = 0, t, 4.0..4.2);
> tA[8] := fsolve(DAA = 0, t, 4.7..4.9);
> tA[9] := fsolve(DAA = 0, t, 5.3..5.5);
> tA[10] := fsolve(DAA = 0, t, 5.5..5.7);
```

$$tA_1 := .3824290257$$
$$tA_2 := .9324776977$$
$$tA_3 := 1.527744928$$
$$tA_4 := 2.341434311$$
$$tA_5 := 2.760135295$$
$$tA_6 := 3.396070545$$
$$tA_7 := 4.072484425$$
$$tA_8 := 4.807203452$$
$$tA_9 := 5.401272765$$
$$tA_{10} := 5.638437107$$

```
> ResultPlot(P[x], P[y], Q[x], Q[y], X(t), Y(t),
>            Dan, 0, 2*Pi, tA, 10, -1.4..1.8, -1.6..2.2);
```

确定正确目标点的第二个可能是使用台球轨迹的参数表示. 这条轨迹包含两条直线段, 从第一个球 P 的位置到球台边缘的目标点 (反射点)T_i 的 l_1 和从 T_i 到第二个球的位置 Q 的 l_2. 这些直线可以使用参数 u 描述为参数形式.

$$l_1: \quad x = P_x + u(T_{i_x} - P_x) \qquad l_2: \quad x = Q_x + u(T_{i_x} - Q_x)$$
$$y = P_y + u(T_{i_y} - P_y) \qquad\qquad y = Q_y + u(T_{i_y} - Q_y)$$

现在我们可以求出直线 l_1 和 l_2 与球台边界的交点. 我们直接计算相应的参数 $u_1, \ldots, u_2$. 如果它们的值满足 $0 < u_i < 1$, 这表明在直线段上至少还有另外一个交点. 于是这条位于球的位置和反射点之间的轨道是不能使用的. 图 7.13 显示出这里有两个不能使用的反射点 T_1 和 T_7. 图中显示出线段 QT_1 与球台的边界在线段的内部相交:

```
> T1[x] := evalf(subs(t = tA[1], X(t))):
> T1[y] := evalf(subs(t = tA[1], Y(t))):
> eq1 := X(t) = Q[x] + u*(T1[x] - Q[x]):
> eq2 := Y(t) = Q[y] + u*(T1[y] - Q[y]):
> fsolve({eq1, eq2}, {u, t}, {u = 0..0.5, t=3*Pi/2..2*Pi});
```

$$\{t = 5.810766136, u = .3288529247\}$$

```
> fsolve({eq1, eq2}, {u, t}, {u = 0.33..0.8, t = 5.82..6.2});
```

$$\{u = .5576715283, t = 6.162750379\}$$

我们看到线段 QT_1 在点 Q 和 T_1 之间与球台的边界相交了两次, 相应的参数值是 $u_1 = 0.3288529247$ 和 $u_2 = 0.5576715283$.

7.5 结论

使用 MAPLE, 我们用两种方法解决了广义台球问题. 图形和数值表明两个方法的结果是相同的. 使用广义反射法得出最终的方程要比最短轨线法更加困难. 但是当所有的变量被替换以后, 第一个方法的最终方程 DA 要比第二个方法的结果 dl 更简单. 使用第二个方法, 我们必须计算各种函数的平方根及其一阶导数. 因此第二个方法的最终方程要更加复杂. 复杂到什么程度要取决于球台边界的形状. 对于某些边界函数完全可能第二个方法的结果更简单.

第一个方法可以被推广到 N 次反射的轨道. 也就是说第一个球在撞击到第二个球之前首先要撞击边界 N 次. 但是第二个方法就没有这样一个推广. 在这个情形下最短轨线的性质不会满足反射定律.

参考文献

[1] W. GANDER and D. GRUNTZ, *The Billiard Problem*, Int. J. Educ. Sci. Technol.,23, 1992, pp. 825-830.
[2] R. G. LERNER and G. L. TRIGG, *Encyclopedia of Physics*, VCH Publishers, New York, 1991.
[3] J. WALDVOGEL, *The Problem of the Circular Billiard*, El. Math., 47, 1992, pp. 108-113.

第八章　镜面曲线

S.Bartoň

8.1　尚未解决的问题

为解决广义台球问题我们使用了广义反射法. 这个方法基于对第一个球的位置的镜像点 M 的计算. 在第七章的图 7.2 中当我们沿着球台的边界移动点 T 时点 M 也跟着移动. M 就描绘出一条镜面曲线, 它依赖于边界在点 T 的切线斜率的变化. 这条镜面曲线依赖于点 P 的位置和球台边界的形状.

在求解广义台球问题时没有必要去确定镜面曲线的形状. 但是这条曲线是非常重要的. 很遗憾还没有进一步讨论过它. 我们将使用 MAPLE 的图形功能来做这件事.

8.2　使用 MAPLE 生成镜面曲线

因为镜像点的位置由两个坐标 $M \equiv [M_x, M_y]$ 给出, 它们都依赖于参数 t, 我们就得到了镜面曲线的参数方程 (参见 [1]). 将微分运算用于第七章得到的方程 DA (参见 7.2.3 和表 7.1) 很容易得到曲线的形状. 这里我们仅仅显示计算 M_x, M_y 的必需的步骤. 结果是

```
> E2 := solve({C[y] - P[y] = k[2]*(C[x] - P[x]),
>              C[y] - Y(t) = k[1]*(C[x] - X(t))}, {C[x], C[y]}):
> assign(E2):
> k[1] := diff(Y(t), t)/diff(X(t),t):
> k[2] := -1/k[1]:
> M[x] := normal(P[x] + 2*(C[x] - P[x]));
> M[y] := normal(P[y] + 2*(C[y] - P[y]));
```

$$
\begin{aligned}
M_x \ := -(&-(\tfrac{\partial}{\partial t}\mathrm{X}(t))^2 P_x + P_x (\tfrac{\partial}{\partial t}\mathrm{Y}(t))^2 - 2 P_y (\tfrac{\partial}{\partial t}\mathrm{Y}(t)) (\tfrac{\partial}{\partial t}\mathrm{X}(t)) \\
&+2\,\mathrm{Y}(t) (\tfrac{\partial}{\partial t}\mathrm{Y}(t)) (\tfrac{\partial}{\partial t}\mathrm{X}(t)) - 2 (\tfrac{\partial}{\partial t}\mathrm{Y}(t))^2 \mathrm{X}(t) \Big/ ((\tfrac{\partial}{\partial t}\mathrm{X}(t))^2 + (\tfrac{\partial}{\partial t}\mathrm{Y}(t))^2)
\end{aligned} \tag{8.1}
$$

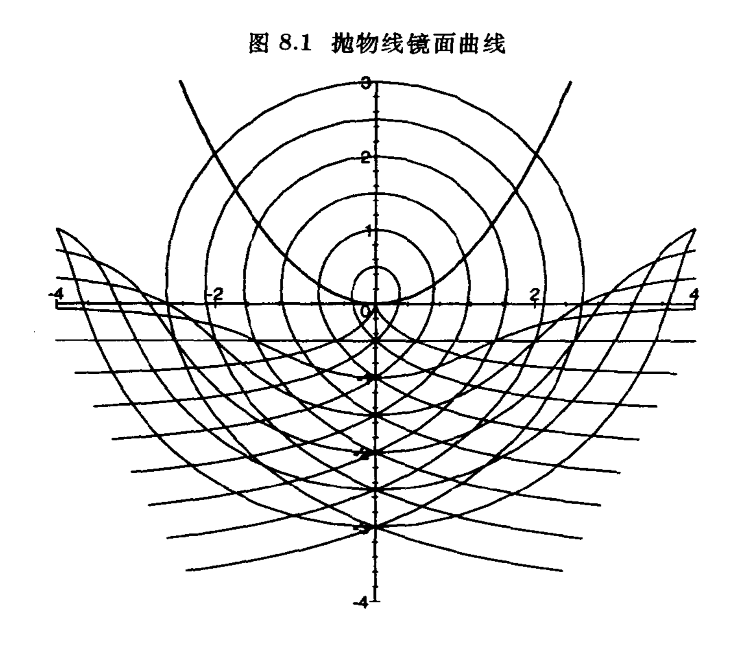
$$
\begin{aligned}
M_y \ := (&-P_y (\tfrac{\partial}{\partial t}\mathrm{X}(t))^2 + P_y (\tfrac{\partial}{\partial t}\mathrm{Y}(t))^2 + 2 (\tfrac{\partial}{\partial t}\mathrm{X}(t))^2 \mathrm{Y}(t) - 2\,\mathrm{X}(t) (\tfrac{\partial}{\partial t}\mathrm{Y}(t)) (\tfrac{\partial}{\partial t}\mathrm{X}(t)) \\
&+ 2 P_x (\tfrac{\partial}{\partial t}\mathrm{Y}(t)) (\tfrac{\partial}{\partial t}\mathrm{X}(t))) \Big/ ((\tfrac{\partial}{\partial t}\mathrm{X}(t))^2 + (\tfrac{\partial}{\partial t}\mathrm{Y}(t))^2)
\end{aligned} \tag{8.2}
$$

对于给定的样板曲线和一个镜像点 P, 我们能够生成它的镜面曲线. 例如, 考虑一条抛物线. 令点 P 沿着 y 轴由初始位置 $P_y = -3$ 到终止位置 $P_y = 3$ 移动, 步长为 $\Delta y = 1$(参见图 8.1). 在三维图上 (图 8.2) 我们可以看到以点 P_y 为函数的镜面曲线的连续变形.

```
> X(t) := t:  Y(t) := t^2/2:
> P[x]:= 0:
> m[x] := simplify(M[x]):  m[y] := simplify(M[y]):
> SPy := [seq(i/2, i = -6..6)]:
> p1 := plot({seq([m[x], m[y], t=-4..4], P[y] = SPy)},
```

```
>             color = black, thickness = 1):
> p2 := plot([X(t), Y(t), t = -4..4], view = [-4 ..  4, -4 ..  4],
>            thickness = 2):
> with(plots):
> display({p1, p2}, scaling = constrained);
> P[y] := 'P[y]':
> plot3d(subs(P[y] = Py,[P[y], m[x], m[y]]), Py = -3..3,
>        t = -3..3, axes = boxed, orientation = [15,30],
>        labels = ['Py', 'Mx', 'My'], grid = [40, 40] );
```

图 8.1 抛物线镜面曲线



8.3 反问题

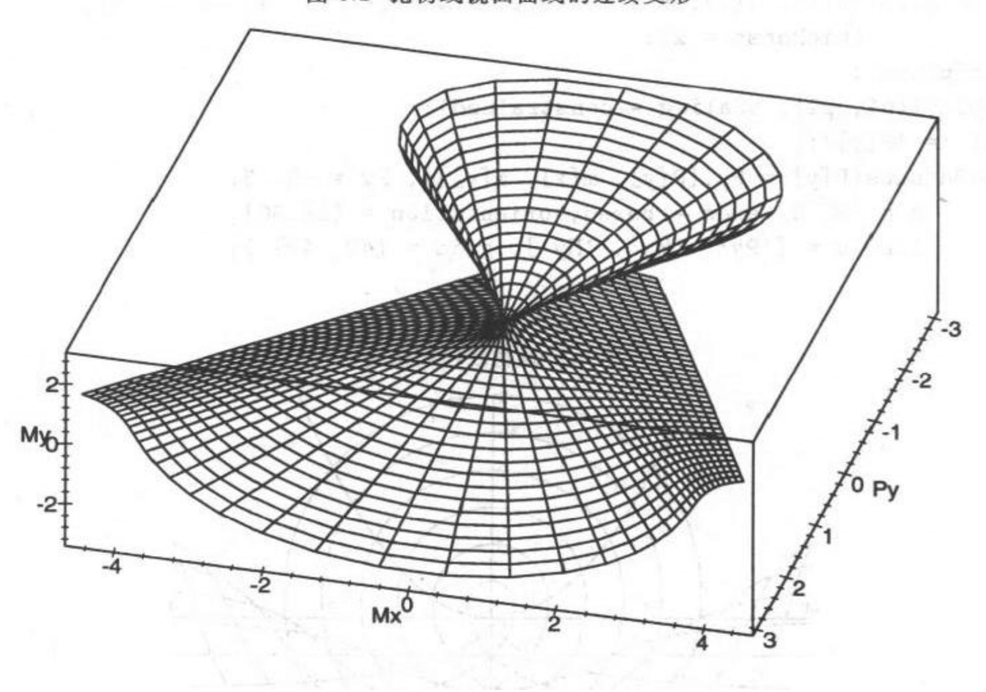
对于已知的镜面曲线和一个给定的点，求原始的样板曲线，这是镜面曲线问题的反问题. 反问题是非常复杂的. 在这个情形下 $M_x = M_x(t), M_y = M_y(t), P_x, P_y$ 是已知的，我们必须对函数 $X(t), Y(t)$ 求解微分方程组 (8.1) 和 (8.2). 一般来说，使用 MAPLE 的 dsolve 函数无法给出微分方程组的解析解，必须使用数值解法.

8.3.1 迂回求解

然而，非常有意思的是使用几何证明我们能够得到微分方程组 (8.1) 和 (8.2) 的准确的解析解. 为了几何地求解这个问题，我们考虑图 8.3. 它说明了对于给定的镜面曲线和镜像点如何求出样板曲线. 令 M_c 是给定的镜像面曲线， P_c 是样板曲线 (原始的曲线), P 是给定的镜像点， M 是镜面曲线上的点， M_1 和 M_2 分别是镜面曲线上在 M 点左边和右边的点， T 是样板曲线上的点， l_t 是曲线 P_c 在点 T 的切线.

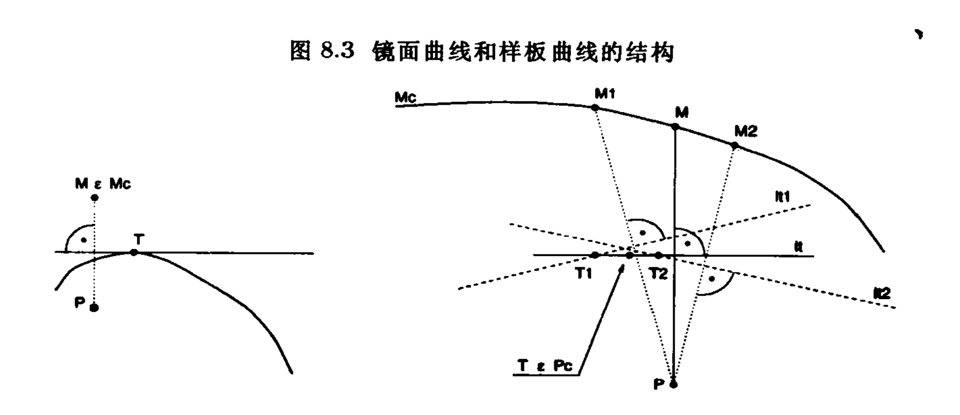
首先我们必须记住镜面曲线 M_c 是如何产生的. 我们选择样板曲线 P_c 上的一点 T, 我们做样

图 8.2 抛物线镜面曲线的连续变形



板曲线在这个点的切线 l_t. 这条线作为镜面直线被用来构成点 M(即 l_t 是点 P 和 M 的对称轴). 当点 T 沿着 P_c 移动时, M 就描绘出了镜面曲线 M_c.

图 8.3 镜面曲线和样板曲线的结构



8.3.2 样板曲线上点的几何结构

给定一个固定的点 P 和一条镜面曲线, 我们的问题是要确定样板曲线. 仅仅求出一个点就足够了. 如果能够求出样板曲线上的一个点, 则就可能一个点一个点地构成整个的曲线. 由上一节的分析可知, 样板曲线的切线是固定点 P 和镜面曲线 M_c 上某一点 M 之间的对称轴. 这在求解反问题中将是有用的.

我们可以在镜面曲线 M_c 上选择点 M, 点 M_1 和 M_2. 令点 M_1 是 M 左侧的一个“小”距离的

点，M_2 是点 M 右侧的点. 则连接点 P 和镜面曲线上的相应点，我们就得到线段 l_c, l_{c_1} 和 l_{c_2}. 令 l_t, l_{t_1}, l_{t_2} 是相应的对称轴. 令 T_1 是直线 l_t 和 l_{t_1} 的交点，同时令 T_2 是直线 l_t 和 l_{t_2} 的交点. 由点 T_1 和 T_2 描出的线段就是所要求的样板曲线的割线. 随着 M_1 和 M_2 趋近于中心点 M, 割线逐渐变短. 有如下极限

$$\lim_{\substack{M_1 \to M^- \\ M_2 \to M^+}} \overleftrightarrow{T_1 T_2} = 0$$

于是这条线段就收敛为一个点. 这个点就是我们要求的样板曲线上的点 P_c. 对于下一个点我们将重复同样的过程来确定.

表 8.1 反问题中使用的主要变量

变量	意义
$X(t), Y(t)$	$X(t) = f(t), Y(t) = g(t)$: 镜面曲线
Ξ, Θ	$\Xi = \varphi(t), \Theta = \psi(t)$: 计算的样板曲线
P_x, P_y	固定点的坐标
$\mathrm{d}t$	Δt, 参数 t 的 “小” 步长
$\mathrm{d}X(t)$	$X(t \pm \Delta t)/\Delta t$
$\mathrm{d}Y(t)$	$Y(t \pm \Delta t)/\Delta t$

8.3.3 MAPLE 解

镜面问题反问题的解析解基于前面给出的几何结构. 因为点 M_1, M_2 处于距中心点 M 很小的距离处，我们可以用 Taylor 级数去表示函数 $M_{1_x}(t), M_{1_y}(t), M_{2_x}(t), M_{2_y}(t)$.

为求解这个反问题我们使用表 8.1 所列的函数和变量.

```
> X(t) := 'X(t)':  Y(t) := 'Y(t)':
> P[x] := 'P[x]':  P[y] := 'P[y]':
> C1[x] := (X1(t) + P[x])/2:
> C1[y] := (Y1(t) + P[y])/2:
> C2[x] := (X2(t) + P[x])/2:
> C2[y] := (Y2(t) + P[y])/2:
> k[1] := -1*(P[x] - X1(t))/(P[y] - Y1(t)):
> k[2] := -1*(P[x] - X2(t))/(P[y] - Y2(t)):
> E1 := solve({Theta - C1[y] = k[1]*(Xi - C1[x]),
>        Theta - C2[y] = k[2]*(Xi - C2[x])}, {Xi, Theta}):
> assign(E1):
> X1(t) := X(t) - dX(t)*dt:  Y1(t) := Y(t) - dY(t)*dt:
> X2(t) := X(t) + dX(t)*dt:  Y2(t) := Y(t) + dY(t)*dt:
> Xi := normal(Xi); Theta := normal(Theta);
```

$$\Xi := -\frac{1}{2}(-P_x{}^2\,\mathrm{dY}(t) - P_y{}^2\,\mathrm{dY}(t) - \mathrm{Y}(t)^2\,\mathrm{dY}(t) + \mathrm{dY}(t)\,\mathrm{X}(t)^2 + \mathrm{dY}(t)\,dt^2\,\mathrm{dX}(t)^2$$
$$- 2\,\mathrm{Y}(t)\,\mathrm{X}(t)\,\mathrm{dX}(t) + 2\,P_y\,\mathrm{X}(t)\,\mathrm{dX}(t) + \mathrm{dY}(t)^3\,dt^2 + 2\,P_y\,\mathrm{Y}(t)\,\mathrm{dY}(t)) \Big/ ($$
$$-P_y\,\mathrm{dX}(t) + P_x\,\mathrm{dY}(t) + \mathrm{Y}(t)\,\mathrm{dX}(t) - \mathrm{dY}(t)\,\mathrm{X}(t))$$

$$\Theta := \frac{1}{2}(-P_y{}^2\,\mathrm{dX}(t) - 2\,\mathrm{X}(t)\,\mathrm{Y}(t)\,\mathrm{dY}(t) + 2\,P_x\,\mathrm{Y}(t)\,\mathrm{dY}(t) + 2\,P_x\,\mathrm{X}(t)\,\mathrm{dX}(t) + \mathrm{dX}(t)^3\,dt^2$$

$$+\,\mathrm{dX}(t)\,\mathrm{Y}(t)^2 + \mathrm{dX}(t)\,dt^2\,\mathrm{dY}(t)^2 - \mathrm{X}(t)^2\,\mathrm{dX}(t) - {P_x}^2\,\mathrm{dX}(t))\Big/(
-P_y\,\mathrm{dX}(t) + P_x\,\mathrm{dY}(t) + \mathrm{Y}(t)\,\mathrm{dX}(t) - \mathrm{dY}(t)\,\mathrm{X}(t))$$

8.3.4 解析解

对于 Ξ, Θ 的 MAPLE 表达式仅仅包含 dt^2 项, 没有 dt 项. 如果计算表达式 $\lim_{dt\to 0}\Xi$ 和 $\lim_{dt\to 0}\Theta$ 则 dt^2 项趋于零, 同时 $\mathrm{dX}(t)$ 和 $\mathrm{dY}(t)$ 项变为 $X_t(t), Y_t(t)$.

```
> Xi := limit(Xi, dt = 0):  Theta := limit(Theta, dt = 0):
> dX(t) := diff(X(t), t):  dY(t) := diff(Y(t), t):
> Xi := collect(Xi, [diff(X(t), t), diff(Y(t) ,t)]);
> Theta := collect(Theta, [diff(X(t), t), diff(Y(t), t)]);
```

$$\Xi := \frac{1}{2}((2\,\mathrm{Y}(t)\,\mathrm{X}(t) - 2\,P_y\,\mathrm{X}(t))\,(\tfrac{\partial}{\partial t}\,\mathrm{X}(t)) + ({P_x}^2 + {P_y}^2 + \mathrm{Y}(t)^2 - \mathrm{X}(t)^2 - 2\,P_y\,\mathrm{Y}(t))
(\tfrac{\partial}{\partial t}\,\mathrm{Y}(t)))\Big/((\mathrm{Y}(t) - P_y)\,(\tfrac{\partial}{\partial t}\,\mathrm{X}(t)) + (P_x - \mathrm{X}(t))\,(\tfrac{\partial}{\partial t}\,\mathrm{Y}(t))) \tag{8.3}$$

$$\Theta := -\frac{1}{2}(({P_y}^2 - 2\,P_x\,\mathrm{X}(t) + {P_x}^2 - \mathrm{Y}(t)^2 + \mathrm{X}(t)^2)\,(\frac{\partial}{\partial t}\,\mathrm{X}(t)) + (2\,\mathrm{Y}(t)\,\mathrm{X}(t) - 2\,P_x\,\mathrm{Y}(t))
(\frac{\partial}{\partial t}\,\mathrm{Y}(t)))\Big/((\mathrm{Y}(t) - P_y)\,(\frac{\partial}{\partial t}\,\mathrm{X}(t)) + (P_x - \mathrm{X}(t))\,(\frac{\partial}{\partial t}\,\mathrm{Y}(t)))$$

使用 `dsolve` 关于 $X(t)$ 和 $Y(t)$ 求解方程组 (8.1) 和 (8.2) 对于 MAPLE 来说一般是非常复杂的. 但是通过我们的迂回求解方法就可以使用 MAPLE 在一般的情形下得到明确的解析解 (8.3).

8.4 例子

现在我们可以使用计算样板曲线的方程 (8.3) 来检验我们的解. 求得的样板曲线的镜面曲线应该与样板曲线计算中作为输入的已知镜面曲线相同.

8.4.1 圆作为镜面曲线

我们可以求出对于圆和一个固定点的样板曲线. 不失一般性, 我们可以假设镜面曲线的半径为 1, 即, 我们使用单位圆作为镜面曲线. 现在对于固定点 P 的每个位置我们可以旋转坐标系使得 $P_y = 0$, 坐标 P_x 可以是变量. 如果 P 位于圆的内部, 则 $P_x < 1$. 如果 $P_x > 1$ 则 P 在圆的外部. 当 $P_x = 1$ 时 P 点在圆上.

```
> X(t) := cos(t):  Y(t) := sin(t):  P[y] := 0:
> xi := simplify(Xi); theta := simplify(Theta);
```

$$\xi := \frac{1}{2}\,\frac{\cos(t)\,(-1 + {P_x}^2)}{-1 + \cos(t)\,P_x}$$

$$\theta := \frac{1}{2}\,\frac{\sin(t)\,(-1 + {P_x}^2)}{-1 + \cos(t)\,P_x}$$

```
> X(t) := xi:  Y(t) := theta:
> m[x] := simplify(M[x]); m[y] := simplify(M[y]);
```

$$m_x := \cos(t)$$

$$m_y := \sin(t)$$

当样板曲线 ξ,θ 被代入方程中计算镜面曲线 $X(t),Y(t)$ 时，输入和输出是相等的. 我们可以用它来证明两个曲线的方程组的由来是正确的.

现在我们必须求出样板曲线的方程. 它的参数式是:

$$x = \frac{\cos(t)(P_x^2-1)}{2(P_x\cos(t)-1)}, y = \frac{\sin(t)(P_x^2-1)}{2(P_x\cos(t)-1)}$$

它看起来似乎是一条圆锥截线. 为确实起见，我们求出这条曲线的解析表达式.

```
> E1 := simplify(subs(t = solve(xi = X, t),
>                     Y^2 = theta^2), symbolic):
> E2 := collect(4*E1, [X^2, X ,P[x]]):
> E3 := factor((E2)):
> E4 := normal(map(t->t/(P[x]^2-1), subs(X = (Xs+P[x])/2, E3))):
> E5 := subs(Xs = 2*X-P[x] ,map(t->t+Xs^2, -expand(E4)));
```

$$E5 := -4\,\frac{Y^2}{-1+{P_x}^2} + (2\,X - P_x)^2 = 1$$

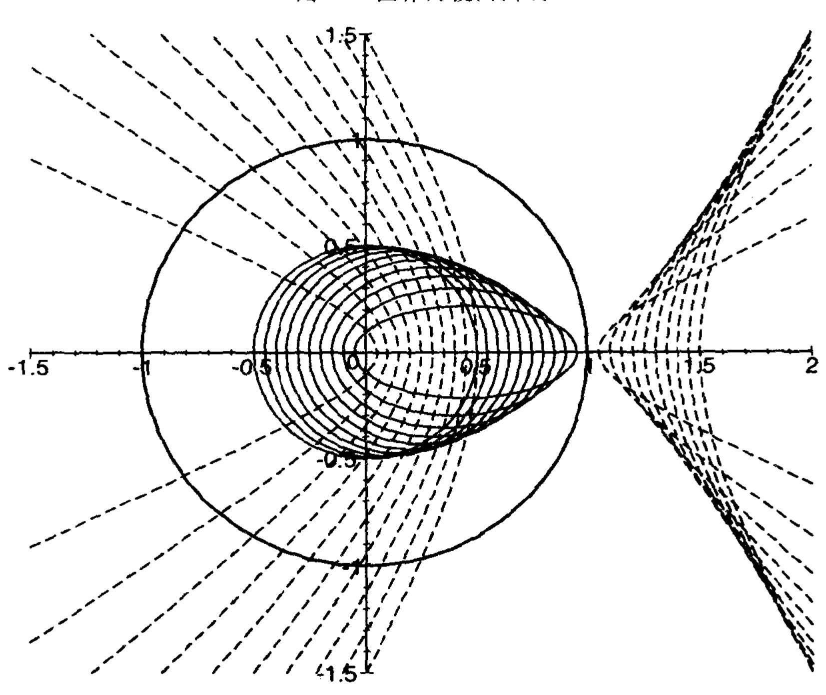
正如我们从最后的方程 $E5$ 所看到的，单位圆的样板曲线总是一条圆锥截线. 如果固定点在圆的内部，这条样板曲线是椭圆. 对于圆外的点它是双曲线. 圆锥的中心 C 总是有坐标 $C_x = P_x/2, C_y = 0$. 圆锥的主半轴是 $a = 1/2$, 同时次半轴是 $b = \sqrt{|P_x^2-1|}/2$. 对于椭圆有离心率 $e = \sqrt{a^2-b^2}$. 对于双曲线有 $e = \sqrt{a^2+b^2}$. 我们可以计算两个焦点之间的距离，它们等于 $2e$ 和 P_x. 因为圆锥的中心有坐标 $C_x = P_x/2$, 焦点的坐标总是等于 $F_1 \equiv [0,0]$ 和 $F_2 \equiv [P_x, 0]$. 我们现在展示几个例子. 为了避免不连续性有必要把图 p2 分为 p2l 和 p2r 两部分.

```
> X(t) := cos(t):  Y(t) := sin(t):
> P[y] := 0:  P[x] := 'P[x]':
> SPx:=[seq(i/10,i=0..9)]:
> p1 := plot({seq([xi, theta, t = 0..2*Pi], P[x] = SPx)},
>             color = black, thickness = 0):
> P[x] := 'P[x]':
> SPx := [seq(1+i/10, i = 1..10)]:
> PxR := arccos(1/P[x])*99/100:
> p2r := plot({seq([xi, theta, t = -PxR..PxR], P[x]=SPx)},
>             color = black, linestyle = 3):
> P[x] := 'P[x]':  PxL := arccos(1/P[x])*101/100:
> p2l := plot({seq([xi, theta, t = PxL..2*Pi - PxL],
>              P[x]=SPx)}, color = black, linestyle = 3):
> p3 := plot([X(t), Y(t), t = 0..2*Pi],
>             view = [-1.5 ..2, -1.5..  1.5], thickness = 2):
> display({p1, p2r, p2l, p3}, scaling = constrained);
```

8.4.2 直线作为镜面曲线

现在我们考虑直线作为镜面曲线. 因为这个问题有移动的对称性，我们可以令镜面直线在坐标系的 X 轴上，而固定点 P 在 Y 轴上，于是 $P_x = 0$ 和 P_y 是变量. 于是我们可以计算 P_y 的样

图 8.4 圆作为镜面曲线



板曲线函数.

```
>  X(t) := t:  Y(t) := 0:  P[x] := 0:  P[y] := 'P[y]':
>  xi := simplify(Xi); theta := simplify(Theta);
```

$$\xi := t$$
$$\theta := \frac{1}{2}\,\frac{{P_y}^2 + t^2}{P_y}$$

```
>  X(t) := xi:  Y(t) := theta:
>  m[x] := simplify(M[x]); m[y] := simplify(M[y]);
```

$$m_x := t$$
$$m_y := 0$$

```
>  E1 := subs(xi = X, Y = theta);
```

$$E1 := Y = \frac{1}{2}\,\frac{{P_y}^2 + X^2}{P_y}$$

我们得到一个依赖于参数 P_y 的抛物线族. 现在我们尝试求出其包络并绘制一些例子.

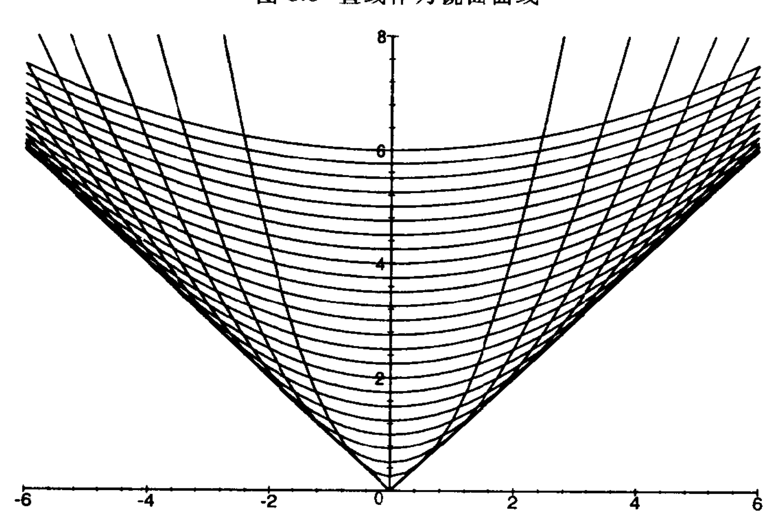
```
>  E2 := Y = X: E3 := Y =- X:
>  solve({E1, E2}, {X,Y}); solve({E1, E3}, {X,Y});
```

$$\{X = P_y, Y = P_y\}, \{X = P_y, Y = P_y\}$$
$$\{Y = P_y, X = -P_y\}, \{Y = P_y, X = -P_y\}$$

我们看到，直线的样板曲线总是一个抛物线方程. 抛物线的顶点是从点 P 到镜面直线的垂线线段的中点. 抛物线的焦点与固定点 P 重合. 当坐标 P_y 改变它的数值时，抛物线充满了由条件 $y \geq x$ 和 $y \geq -x$ 给出的区域. 这两条直线构成了抛物线的包络. 图 8.5 显示出了这些抛物线的边界.

```
> SPy := [seq(i/4, i = 1..16)]:
> p1 := plot({seq([xi, theta, t = -6..6], P[y] = SPy)},
>               color = black, thickness = 1):
> p2 := plot([[-6, 6], [0, 0], [6, 6]], thickness = 2):
> display({p1, p2}, view = [-6..6, 0..8], scaling = constrained);
```

图 8.5 直线作为镜面曲线



8.5 结论

使用 MAPLE 成功地解决了镜面问题的反问题. 完全可能求出圆和直线的以固定点 P 为函数的样板曲线的一般形状. 为此必须求解一个很大的方程组，即 (8.1), (8.2) 和 (8.3), 这是毫无困难的.

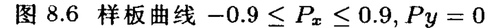


图 8.6 样板曲线 $-0.9 \leq P_x \leq 0.9, Py = 0$

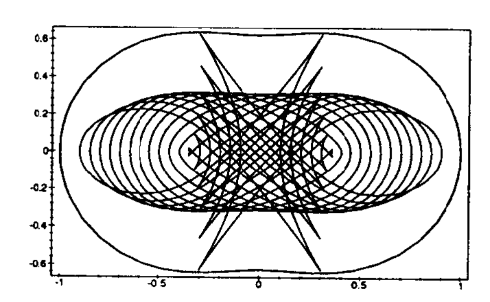
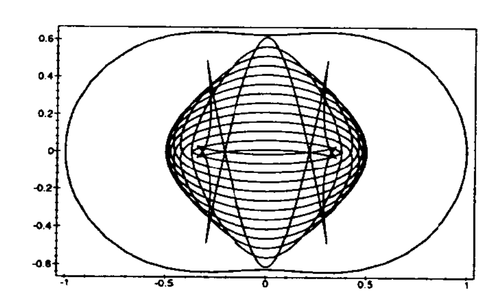


图 8.7 样板曲线 $P_x = 0, -0.6 \leq P_y \leq 0.6$



我们可以展示以固定 (镜像) 点为函数的样板曲线的连续变形 (参见图 8.6 和 8.7). 如果我们沿着任意的一条适当的曲线移动固定点，同时使得相应于固定点位置的镜面曲线位于垂直于已知曲

线的平面上．这个平面与给定曲线的交点与固定点的瞬时位置相重合．使用位移长度作为第三坐标，我们可以产生图 8.8 所示的三维图形．

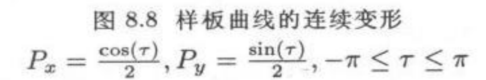
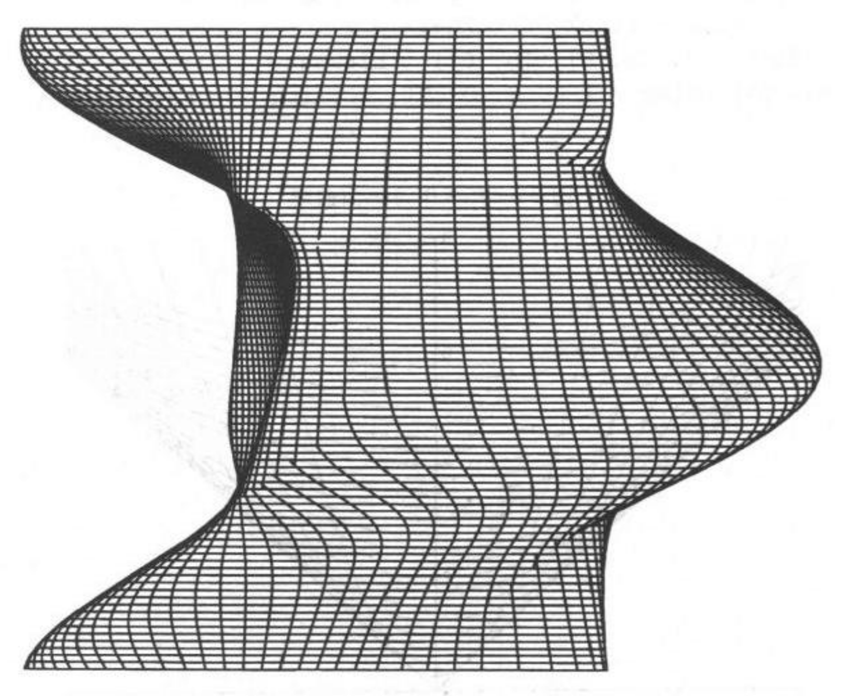


图 8.8 样板曲线的连续变形

$P_x = \frac{\cos(\tau)}{2}, P_y = \frac{\sin(\tau)}{2}, -\pi \le \tau \le \pi$



```
> a:=1:  b:=5/8:
> rho := sqrt(a^2*cos(t)^2 + b^2*sin(t)^2):
> X(t) := rho*cos(t):  Y(t) := rho*sin(t):
> p1 := plot([X(t), Y(t), t = -Pi..Pi], thickness = 3):
> P[x] := 'P[x]':  P[y] := 0:
> xi := simplify(Xi,symbolic):  theta := simplify(Theta,symbolic):
> SPx := [seq(i/10, i = -9..9)]:
> p2 := plot({seq([xi, theta, t = -Pi..Pi], P[x] = SPx)},
>             thickness = 1, color = black):
> display({p1, p2}, axes = boxed, scaling = constrained);
> P[x] := 0:  P[y] := 'P[y]':
> xi := simplify(Xi,symbolic):  theta := simplify(Theta, symbolic):
> SPy := [seq(i/10, i =-6..6)]:
> p2 := plot({seq([xi, theta, t= -Pi..Pi],P[y] = SPy)},
>             thickness = 1, color = black):
> display({p1, p2}, axes = boxed, scaling = constrained);
> P[x] := cos(tau)/2:  P[y]:=sin(tau)/2:
> xi := simplify(Xi, symbolic):  theta := simplify(Theta, symbolic):
> plot3d([xi, theta, tau] ,t = -Pi..Pi, tau = -Pi..Pi ,
>           grid = [90, 90], axes = none, orientation = [-110, 90]);
```

我们还不知道镜面曲线在现实世界的应用中是不是有用. 似乎由我们的几何讨论所得到的解析解是全新的. 我们可以想象到它在物理学特别是在光学上的应用. 例如, 对于已知的光学仪器上由样板曲线描述的光学性质, 人们就可以求出由镜面曲线描述的技术参数.

参考文献

[1] H.J. BARTSCH, *Taschenbuch Mathematischer Formeln*, Fachbuchverlag, BRD, Leipzig,1991.

第九章　光滑滤子

W. Gander, U. von Matt

9.1　引言

在许多的应用中，有一个应用是测量变量. 变量本身缓慢地变化，同时受到随机噪声的干扰. 于是，为了重建潜在的光滑函数，通常应用一个光滑滤子来测量数据. 假设噪声是独立于观察变量的，并假设噪声服从均值为 0, 标准差为 δ 的正态分布.

在本章中，我们将讨论这种光滑问题的两种不同方法： Savitzky-Golay 滤子和最小二乘滤子. 借助测试函数

$$F(x) := \mathrm{e}^{-100(x-1/5)^2} + \mathrm{e}^{-500(x-2/5)^2} + \mathrm{e}^{-2500(x-3/5)^2} + \mathrm{e}^{-12500(x-4/5)^2}, \tag{9.1}$$

我们将分析这两个滤子的性质. 此函数有四个不同宽度的凸起 (参见图 9.1).

在 MATLAB 中，由算法 9.1 的语句，可产生一个长度 $n = 1000$ 的向量 $\mathbf{f}$, 它由测试数据组成，此数据受到标准差 $\delta = 0.1$ 的随机噪声的干扰. 我们得到图 9.2 所显示的样本数据.

在下面几节里，我们将用 $f_i, i = 1, \ldots, n$ 表示测试数据，用 $g_i, i = 1, \ldots, n$ 表示光滑数据.

算法 9.1 噪声数据的生成

```
n = 1000;
delta = 0.1;
x = [0:n-1]'/(n-1);
F = exp (- 100*(x - 1/5).^2) + exp (-  500*(x - 2/5).^2) + ...
    exp (-2500*(x - 3/5).^2) + exp (-12500*(x - 4/5).^2);
randn ('seed', 0);
f = F + delta * randn (size (x));
```

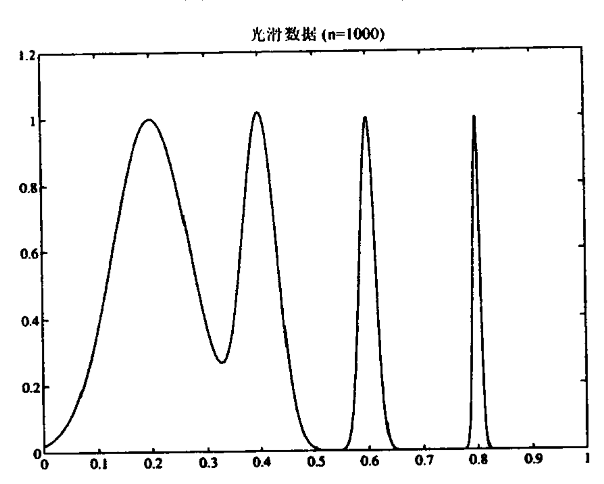
9.2　Savitzky-Golay 滤子

在 1964 年， A. Savitzky 和 M. J. E. Golay [10] 引入这种光滑方法. 原始论文有一些错误，但在 [11] 中已被改正. 读者也能在 [8] 中找到有关此主题的另一些介绍.

主要思想是，通过取点 x_i 附近数据的平均，得到它的光滑数值 g_i. 最简单的方法是计算固定数目的 f_i 的移动平均.

更一般地，我们也可根据这固定个数的点拟合一个多项式. 多项式在 x_i 的值，就给出光滑值 g_i. 图 9.3 显示这种思想，其中 n_L 表示 x_i 左边点的个数和 n_R 表示 x_i 右边点的个数， $p_i(x)$ 表示一个 M 次多项式，它在最小二乘意义下拟合这 $n_L + n_R + 1$ 个点，因此，有 $g_i = p_i(x_i)$.

图 9.1 光滑函数 $F(x)$



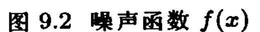
9.2.1 滤子系数

根据数据 f_i 拟合的 M 次多项式 $p_i(x)$, 可写成

$$p_i(x) := \sum_{k=0}^{M} b_k \left(\frac{x - x_i}{\Delta x}\right)^k . \tag{9.2}$$

假设横坐标 x_i 具有 $x_{i+1} - x_i \equiv \Delta x$ 的均匀间距. 为了在最小二乘意义下用 $p_i(x)$ 拟合测试数据, 我们必须确定系数 b_k, 使得

$$\sum_{j=i-n_L}^{i+n_R} (p_i(x_j) - f_j)^2 = \min . \tag{9.3}$$


图 9.2 噪声函数 $f(x)$

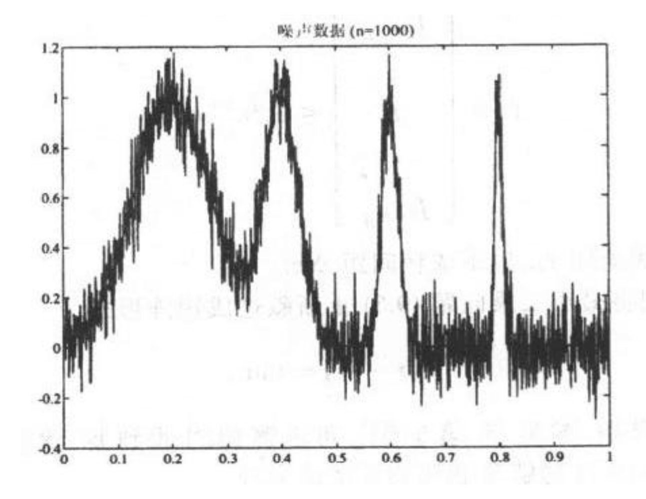
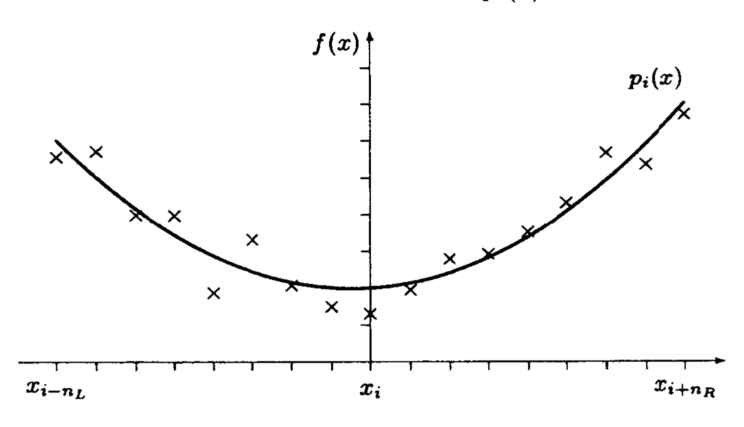



图 9.3 最小二乘多项式 $p_i(x)$



我们定义矩阵

$$A := \begin{bmatrix} (-n_L)^M & \cdots & -n_L & 1 \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & 1 \\ \vdots & & \vdots & \vdots \\ n_R^M & \cdots & n_R & 1 \end{bmatrix} \in \mathbf{R}^{(n_L+n_R+1)\times(M+1)} \tag{9.4}$$

及两个向量

$$\mathbf{b} := \begin{bmatrix} b_M \\ \vdots \\ b_1 \\ b_0 \end{bmatrix} \in \mathbf{R}^{M+1} \tag{9.5}$$

和

$$\mathbf{f} := \begin{bmatrix} f_{i-n_L} \\ \vdots \\ f_i \\ \vdots \\ f_{i+n_R} \end{bmatrix} \in \mathbf{R}^{n_L+n_R+1}. \tag{9.6}$$

注意，矩阵 A 既不依赖横坐标 x_i，也不依赖间距 Δx.

利用这些定义，我们将最小二乘问题 (9.3) 重新叙述成矩阵形式

$$\|A\mathbf{b} - \mathbf{f}\|_2 = \min. \tag{9.7}$$

现在利用 A 的 QR 分解 (参见 [5, 第 5 章]), 可求解 (9.7) 得到 $\mathbf{b}$. 然而，为了过滤的目的，只需要知道 $g_i = p_i(x_i) = b_0$.(9.7) 的解 $\mathbf{b}$ 也可表示成法方程

$$A^{\mathrm{T}}A\mathbf{b} = A^{\mathrm{T}}\mathbf{f} \tag{9.8}$$

的解. 于是, 我们得到

$$g_i = \mathbf{e}_{M+1}^{\mathrm{T}}(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}\mathbf{f}, \tag{9.9}$$

其中 $\mathbf{e}_{M+1}$ 是 $(M+1)$ 维的单位向量.

显然, g_i 可表示为 f_i 的一个线性组合. 定义向量

$$\mathbf{c} := A(A^{\mathrm{T}}A)^{-1}\mathbf{e}_{M+1} \tag{9.10}$$

包括滤子系数 $c_{-n_L}, \ldots, c_{n_R}$. 因为 $\mathbf{c}$ 不依赖于 x_i 和 Δx, 只需计算它一次. 利用简单的数乘, 可计算所有的光滑数值 g_i

$$g_i = \mathbf{c}^{\mathrm{T}}\mathbf{f} = \sum_{j=i-n_L}^{i+n_R} c_{j-i}\, f_j. \tag{9.11}$$

对于很大的 M, 根据方程 (9.10). 计算向量 $\mathbf{c}$ 可能不精确. 这种不精确归因于 A 的条件数是 $A^{\mathrm{T}}A$ 的平方. 另一方面, 众所周知, 利用 QR 分解

$$A = QR \tag{9.12}$$

求解最小二乘问题 (9.7) 是稳定的. Q 表示一个 $(n_L+n_R+1)\times(M+1)$ 的正交矩阵, R 表示一个 $(M+1)\times(M+1)$ 的上三角矩阵. 如果将分解 (9.12) 代入 (9.10), 可得到向量 $\mathbf{c}$ 的表达式

$$c = \frac{1}{r_{M+1,M+1}}Q\mathbf{e}_{M+1}. \tag{9.13}$$

这是计算 $\mathbf{c}$ 的一个不错的数值方法. 奇怪的是 [8] 和 [10] 都没有指出它.

9.2.2 结论

算法 9.2 给出了 Savitzky-Golay 光滑滤子. 如果将此算法应用到图 9.2 的光滑问题, 则得到图 9.4 中的光滑曲线. 为了参考起见, 我们也重叠方程 (9.1) 的函数 F 的图形. 对于这个测试情况, 取参数 $n_L = n_R = 16$ 和 $M = 4$ 是最优的. 在一台 Pentium Pro 200MHz 的 PC 机上运行, 算法 9.2 的执行时间是 6.41 ms CPU 时间.

Savitzky-Golay 滤子的主要优点是它的速度. 对于给定 n_L, n_R 和 M 的值, 滤子参数 $\mathbf{c}$ 只需计算一次. 于是, 可用 n_L+n_R+1 长度简单的数乘 (9.11), 计算每个滤子化的值 g_i. 对于特殊的应用, 可以在硬件中实现此运算.

选取滤子参数 n_L, n_R, M 不明显是它的缺点. [1, 8, 13] 给出一些实用的提示. 但是, 在许多情况下, 为了得到最好的结果, 需要一些可视的优化技术.

算法 9.2 Savitzky-Golay 光滑滤子

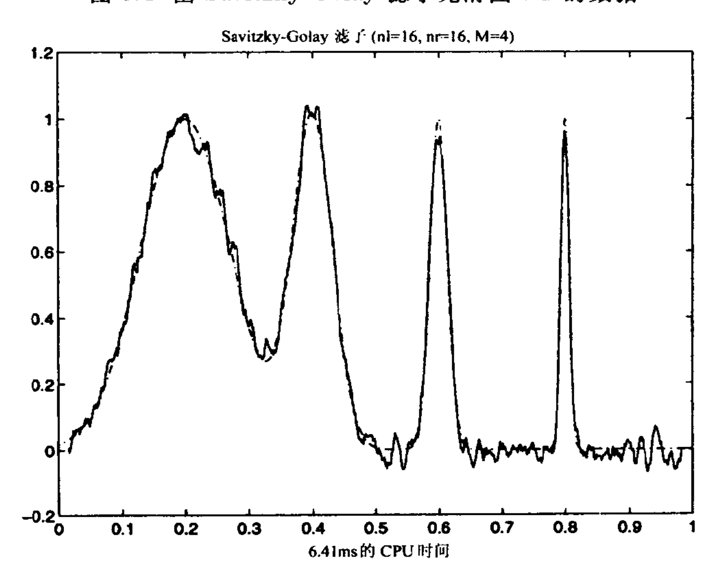
```
function g = SavGol (f, nl, nr, M)

A = ones (nl+nr+1, M+1);
for j = M:-1:1,
  A (:, j) = [-nl:nr]' .* A (:, j+1);
end
[Q, R] = qr (A);
```

```
c = Q (:, M+1) / R (M+1, M+1);

n = length (f);
g = filter (c (nl+nr+1:-1:1), 1, f);
g (1:nl) = f (1:nl);
g (nl+1:n-nr) = g (nl+nr+1:n);
g (n-nr+1:n) = f (n-nr+1:n);
```

图 9.4 由 Savitzky-Golay 滤子光滑图 9.2 的数据



最后，边界点仍是一个问题．它们不能被一个 $n_L > 0$ 或 $n_R > 0$ 的滤子光滑化．正如图 9.4 所示，这些边界值被丢弃，或者对这些特殊情况，构造有 $n_L = 0$ 或 $n_R = 0$ 的特殊的 Savitzky-Golay 滤子．

9.3 最小二乘滤子

对于过滤问题，另一种方法是要求过滤曲线 $g(x)$ 尽可能光滑．在连续的情况下，我们要求

$$\int_{x_{min}}^{x_{max}} g''(x)^2 \mathrm{d}x = \min. \tag{9.14}$$

因为函数 $f(x)$ 只在离散点 x_i 有测量数据 f_i，所以只能在相同的点计算光滑数值 g_i．因此，必须用有限差分方法表示 $g(x)$ 的二阶导数，一个常用的取法为

$$g''(x_i) \approx \frac{g_{i+1} - 2g_i + g_{i-1}}{\Delta x^2}. \tag{9.15}$$

对于间距不等的横坐标 x_i，可类似地给出．因此用条件

$$\sum_{i=2}^{n-1} (g_{i+1} - 2g_i + g_{i-1})^2 = \min. \tag{9.16}$$

代替条件 (9.14). 除此光滑条件之外，还要求在重叠噪声的极限内，用函数 $g(x)$ 近似 $f(x)$. 如果假设 f_i 的值是受到均值为 0, 标准差为 δ 的随机噪声所干扰，则我们要求平均

$$|g_i - f_i| \le \delta. \tag{9.17}$$

对于 n 个样本，此条件可写成

$$\sum_{i=1}^{n} (g_i - f_i)^2 \le n\delta^2. \tag{9.18}$$

现在定义矩阵

$$A := \begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix} \in \mathbf{R}^{(n-2)\times n}. \tag{9.19}$$

于是，可用矩阵符号重述优化问题 (9.16, 9.18) 为，极小化

$$\|A\mathbf{g}\|_2 = \min \tag{9.20}$$

服从二次不等式约束

$$\|\mathbf{g} - \mathbf{f}\|_2^2 \le \alpha^2 := n\delta^2. \tag{9.21}$$

9.3.1 Lagrange 方程

现在研究优化问题 (9.20, 9.21) 的解. 首先，假设矩阵 A 的零空间 $\mathcal{N}(A)$ 中，存在满足约束 (9.21) 的向量. 在这种情况下，要确定满足约束 (9.21) 的最好向量 $\mathbf{g}$, 即考虑极小问题

$$\|\mathbf{g} - \mathbf{f}\|_2 = \min \tag{9.22}$$

服从线性等式约束

$$A\mathbf{g} = \mathbf{0}. \tag{9.23}$$

容易证明，这种情况的唯一最优解是

$$\mathbf{g} = \mathbf{f} - A^{\mathrm{T}}\mathbf{z}, \tag{9.24}$$

其中 $\mathbf{z}$ 是线性最小二乘问题

$$\|A^{\mathrm{T}}\mathbf{z} - \mathbf{f}\|_2 = \min \tag{9.25}$$

的唯一解. 向量 $\mathbf{z}$ 也是对应 (9.25) 的法方程

$$AA^{\mathrm{T}}\mathbf{z} = A\mathbf{f} \tag{9.26}$$

的解.

现在假设零空间 $\mathcal{N}(A)$ 中，不存在满足约束 (9.21) 的向量. 实际上，这意味着由方程 (9.24) 和 (9.25) 定义的向量 $\mathbf{g}\in \mathcal{N}(A)$, 满足

$$\|\mathbf{g} - \mathbf{f}\|_2 > \alpha. \tag{9.27}$$

我们通过引进 Lagrange 函数

$$\Phi(\mathbf{g}, \lambda, \mu) := \mathbf{g}^{\mathrm{T}}A^{\mathrm{T}}A\mathbf{g} + \lambda(\|\mathbf{g} - \mathbf{f}\|_2^2 + \mu^2 - \alpha^2), \tag{9.28}$$

研究最小二乘问题 (9.20, 9.21), 其中 μ 是松弛变量. 求 Φ 关于 $\mathbf{g}, \lambda, \mu$ 微分, 可得到 Lagrange 方程

$$\begin{aligned} (A^{\mathrm{T}}A + \lambda I)\mathbf{g} &= \lambda \mathbf{f}, & (9.29)\\ \|\mathbf{g}-\mathbf{f}\|_2^2 + \mu^2 &= \alpha^2, & (9.30)\\ \lambda\mu &= 0. & (9.31) \end{aligned}$$

对于 $\lambda = 0$, 因为矩阵 A 是满秩, 所以有 $A^{\mathrm{T}}A\mathbf{g} = 0$ 或 $A\mathbf{g} = 0$. 但是, 根据上面的假设, 向量 $\mathbf{g}$ 不是零空间 $\mathcal{N}(A)$ 的向量, 因此从现在开始, 我们假设 $\lambda \neq 0$.

由于 $\lambda \neq 0$, 从方程 (9.31) 可得 $\mu = 0$. 因此简化 Lagrange 方程 (9.29, 9.30, 9.31) 为

$$\begin{aligned} (A^{\mathrm{T}}A + \lambda I)\mathbf{g} &= \lambda \mathbf{f}, & (9.32)\\ \|\mathbf{g}-\mathbf{f}\|_2 &= \alpha^2. & (9.33) \end{aligned}$$

因为 $\lambda \neq 0$, 利用方程 (9.32) 表示 $\mathbf{g}$

$$\mathbf{g} = \mathbf{f} - A^{\mathrm{T}}\mathbf{z}, \tag{9.34}$$

其中

$$\mathbf{z} = \frac{1}{\lambda}A\mathbf{g}. \tag{9.35}$$

将 $\mathbf{g}$ 的表达式 (9.34) 代入 (9.32, 9.33), 可得对偶 Lagrange 方程

$$\begin{aligned} (AA^{\mathrm{T}} + \lambda I)\mathbf{z} &= A\mathbf{f}, & (9.36)\\ \|A^{\mathrm{T}}\mathbf{z}\|_2 &= \alpha. & (9.37) \end{aligned}$$

只要确定 λ 和 $\mathbf{z}$, 就可根据方程 (9.34) 计算 $\mathbf{g}$.

[3, 4] 证明了存在唯一的 Lagrange 乘子 $\lambda > 0$, 和对偶 Lagrange 方程 (9.36, 9.37) 的解 $\mathbf{z}$, 而且由 (9.34) 得到的 $\mathbf{g}$, 是最小二乘问题 (9.20, 9.21) 的解.

对偶 Lagrange 方程 (9.36, 9.37) 的优点是, 对 $\lambda \geq 0$, 矩阵 $AA^{\mathrm{T}} + \lambda I$ 是非奇异的. 通过求解非线性特征方程

$$s(\lambda) := \left\|A^{\mathrm{T}}(AA^{\mathrm{T}} + \lambda I)^{-1}Af\right\|_2^2 = \alpha^2, \tag{9.38}$$

可得到 Lagrange 乘子 λ. 这将在第 9.3.2 节详细讨论.

从而, 我们给出算法 9.3, 它用来求约束最小二乘问题 (9.20, 9.21) 的解. 下一节将集中在用 MATLAB 实现实际问题上.

算法 9.3 求解约束最小二乘问题 (9.20,9.21)

Solve the linear least squares problem (9.25) for $\mathbf{z}$.
if $\|A^{\mathrm{T}}\mathbf{z}\|_2 > \alpha$ **then**
  Solve the secular equation (9.38) for the unique zero $\lambda > 0$.
  Solve the linear system (9.36) for $\mathbf{z}$.
end
$\mathbf{g} := \mathrm{f} - A^{\mathrm{T}}\mathbf{z}$

9.3.2 零点探测器

算法 9.3 的主要任务是，求特征方程 (9.38) 的唯一零点 $\lambda > 0$. 图 9.5 显示特征函数 $s(\lambda)$ 的图形. 因为 $s(\lambda)$ 是非线性函数，所以需要迭代法求解特征方程 (9.38). 由迭代

$$\lambda_{k+1} := \lambda_k - \frac{s(\lambda_k) - \alpha^2}{s'(\lambda_k)} \tag{9.39}$$

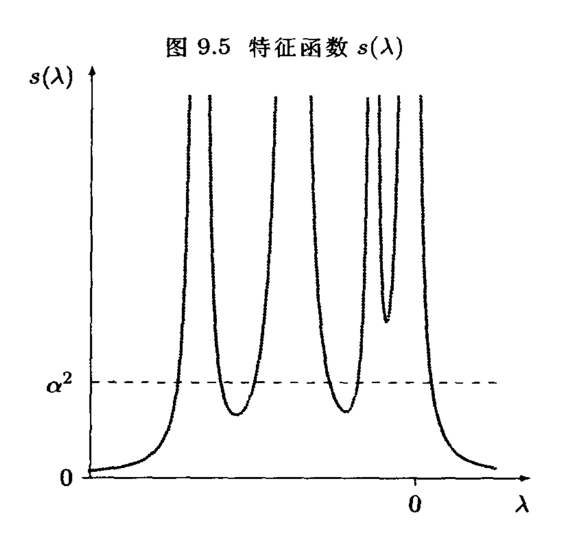
定义的 Newton 法是一个好的选择. 然而，对于我们特殊的方程， Reinsch [9] 提出加速 Newton 迭代法

$$\lambda_{k+1} := \lambda_k - 2\frac{s(\lambda_k)}{s'(\lambda_k)}(\frac{\sqrt{s(\lambda_k)}}{\alpha} - 1). \tag{9.40}$$

如果此迭代从 $\lambda_0 = 0$ 开始，则得到 λ_k 的严格递增序列. 在 [9] 和 [12, 第 65-66 页] 中有这个关键性质的证明. 然而在浮点算法中，不能保持此数学性质. 这种观察导致数值终止准则

$$\lambda_{k+1} \leq \lambda_k. \tag{9.41}$$

图 9.5 特征函数 $s(\lambda)$



9.3.3 特征函数值

为了用迭代 (9.40) 求解特征方程 (9.38), 我们需要一种数值方法求解特征函数 $s(\lambda)$ 和它的导数 $s'(\lambda)$. $s(\lambda)$ 和 $s'(\lambda)$ 的值可表示成

$$\begin{aligned} s(\lambda) &= \|A^{\mathrm{T}}\mathbf{z}\|_2^2, & (9.42)\\ s'(\lambda) &= -2\mathbf{z}^{\mathrm{T}}(\mathbf{z} + \lambda\mathbf{z}'), & (9.43) \end{aligned}$$

其中 $\mathbf{z}$ 和 $\mathbf{z}'$ 满足方程

$$(AA^{\mathrm{T}} + \lambda I)\mathbf{z} = A\mathbf{f} \tag{9.44}$$

和

$$(AA^{\mathrm{T}} + \lambda I)\mathbf{z}' = -\mathbf{z}. \tag{9.45}$$

从方程 (9.44) 和 (9.45) 可知，我们必须求解含有矩阵 $AA^{\mathrm{T}}+\lambda I$ 的线性系统. 然而，我们也可将方程 (9.44), 看成是对应线性最小二乘问题

$$\left\| \begin{bmatrix} A^{\mathrm{T}} \\ \sqrt{\lambda} I \end{bmatrix} \mathbf{z} - \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \right\|_2 = \min \tag{9.46}$$

的法方程. 如果 $\lambda > 0$, 则可类似地计算最小二乘问题

$$\left\| \begin{bmatrix} A^{\mathrm{T}} \\ \sqrt{\lambda} I \end{bmatrix} \mathbf{z}' - \begin{bmatrix} \mathbf{0} \\ -\mathbf{z}/\sqrt{\lambda} \end{bmatrix} \right\|_2 = \min \tag{9.47}$$

的解 $\mathbf{z}'$. 注意，对于 $\lambda = 0$, 在方程 (9.43) 中，不需要向量 $\mathbf{z}'$ 计算 $s'(\lambda)$. 在数值方面，我们喜欢从两个线性最小二乘问题 (9.46, 9.47) 计算 $\mathbf{z}$ 和 $\mathbf{z}'$, 而不是解两个线性系统 (9.44, 9.45).

图 9.6 QR 分解 (9.48) 的第一步

$$\begin{bmatrix} \textcircled{1} & & & & \\ -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ \textcircled{\sigma} & & & & \\ & \sigma & & & \\ & & \sigma & & \\ & & & \sigma & \\ & & & & \ddots \end{bmatrix} \mapsto \begin{bmatrix} \textcircled{r_{11}} & & & & \\ \textcircled{-2} & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ 0 & & & & \\ & \sigma & & & \\ & & \sigma & & \\ & & & \sigma & \\ & & & & \ddots \end{bmatrix} \mapsto$$

$$\begin{bmatrix} \textcircled{r_{11}} & r_{12} & & & \\ 0 & r_{22} & & & \\ \textcircled{1} & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ 0 & & & & \\ & \sigma & & & \\ & & \sigma & & \\ & & & \sigma & \\ & & & & \ddots \end{bmatrix} \mapsto \begin{bmatrix} r_{11} & r_{12} & r_{13} & & \\ 0 & r_{22} & & & \\ 0 & t & r_{33} & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ 0 & & & & \\ & \sigma & & & \\ & & \sigma & & \\ & & & \sigma & \\ & & & & \ddots \end{bmatrix}$$

我们可以借助 QR 分解

$$\begin{bmatrix} A^{\mathrm{T}} \\ \sqrt{\lambda} I \end{bmatrix} = Q \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{9.48}$$

其中 Q 和 R 分别是 $(2n-2)\times(2n-2)$ 的正交矩阵和 $(n-2)\times(n-2)$ 上三角矩阵，求解两个最小二乘问题 (9.46, 9.47). 在 MATLAB 中，用语句

```
>> [Q, R] = qr ([A'; sqrt(lambda)*eye(n-2)]);
```

计算此分解.

然而，此命令没有利用矩阵的特殊结构，并且它的数值复杂性随 A 的大小立方地增加. 我们能设计一个算法，计算 QR 分解 (9.48), 它的数值复杂性随 A 的大小只线性地增加. 我们用一个适当的基变换序列实现这个改进. 当处理第一列时，图 9.6 显示前三个基旋转的应用. σ 是 $\sqrt{\lambda}$ 的简写. 如果应用此步 $n-2$ 次，则得到 QR 分解 (9.48). 算法 9.6 用伪代码显示同样的过程. 通过调用 BLAS 程序 rotg 和 rot, 描述基旋转的构造和应用. [2, 6] 给出它们的精确定义. 对 MATLAB 或一种普通的程序设计语言，如 C 或 Fortran, 实际的实现是简单的.

我们强调，算法 9.6 计算的矩阵 $\bar{Q}$ 与 QR 分解 (9.48) 的矩阵 Q 是不同的. 矩阵 $\bar{Q}$ 反而包含有关基旋转的信息，应用矩阵 Q 于向量 $\mathbf{x}$ 需要这些基旋转. 算法 9.4 和 9.5 给出积 $\mathbf{y}:=Q\mathbf{x}$ 和 $\mathbf{y}:=Q^{\mathrm{T}}\mathbf{x}$ 的值.

算法 9.4 计算乘积 $\mathbf{y}:=Q\mathbf{x}$

$\mathbf{y}:=\mathbf{x}$
for $k:=n-2$ **to** 1 **by** -1 **do**
  rot $(y_k, y_{k+2}, \bar{Q}_{k5}, -\bar{Q}_{k6})$
  rot $(y_k, y_{k+1}, \bar{Q}_{k3}, -\bar{Q}_{k4})$
  rot $(y_k, y_{n+k}, \bar{Q}_{k1}, -\bar{Q}_{k2})$
end

算法 9.5 计算乘积 $\mathbf{y}=Q^{\mathrm{T}}\mathbf{x}$

$\mathbf{y}:=\mathbf{x}$
for $k:=1$ **to** $n-2$ **do**
  rot $(y_k, y_{n+k}, \bar{Q}_{k1}, -\bar{Q}_{k2})$
  rot $(y_k, y_{k+1}, \bar{Q}_{k3}, -\bar{Q}_{k4})$
  rot $(y_k, y_{k+2}, \bar{Q}_{k5}, -\bar{Q}_{k6})$
end

9.3.4 MEX 文件

因为 MATLAB 是一种解释性语言，它不适应运行算法 9.4, 9.5 和 9.6. 由解释器引入的总开销是相当大的. 用一种普通的程序设计语言，如 C 或 Fortran, 我们的程序反而执行得更好. MATLAB 提供一种方法，分开执行编译的程序，即所谓的 MEX 文件.

假设在 C 语言中，我们已经实现算法 9.4, 9.5 和 9.6. 如果我们想在 MATLAB 中执行这些程序，则对每一个程序都需要一个接口. 在 MATLAB 中，函数调用的一般语法为

$$[l_1, l_2, \ldots, l_{\mathrm{nlhs}}] = \mathtt{fct}(r_1, r_2, \ldots, r_{\mathrm{nrhs}});$$

我们称 l_i 为输出参数，称 r_i 为输入参数. 如果用一个外部的 C 子程序实现此函数，则我们需要下面的接口：

```
#include "mex.h"
void mexFunction (int nlhs, mxArray *plhs[],
                  int nrhs, mxArray *prhs[])
{
   ...
}
```

两个参数 nlhs 和 nrhs 给出在 MATLAB 中调用 fct 的左边参数个数和右边参数个数. 参数 plhs 是一个长度为 nlhs 的数组指针, 为了返回的左边的矩阵, 我们必须用指针. 同样地, 参数 prhs 是一个长度为 nrhs 的数组指针, 它的入口指向右边的矩阵.

算法 9.6 计算 QR 分解 (9.48)

Allocate the $(n-2)$-by-6 matrix $\bar{Q}$.
$r_{11} := 1$
$t := -2$
if $n > 3$ **then**
  $r_{22} := 1$
end
for $k := 1$ **to** $n-2$ **do**
  tmp $:= \sqrt{\lambda}$
  rotg$(r_{kk}, \text{tmp}, \bar{Q}_{k1}, \bar{Q}_{k2})$
  rotg$(r_{kk}, t, \bar{Q}_{k3}, \bar{Q}_{k4})$
  if $k < n-2$ **then**
    $r_{k,k+1} := 0$
    rot$(r_{k,k+1}, r_{k+1,k+1}, \bar{Q}_{k3}, \bar{Q}_{k4})$
  end
  tmp $:= 1$
  rotg$(r_{kk}, \text{tmp}, \bar{Q}_{k5}, \bar{Q}_{k6})$
  if $k < n-2$ **then**
    $t := -2$
    rot$(r_{k,k+1}, t, \bar{Q}_{k5}, \bar{Q}_{k6})$
    if $k < n-3$ **then**
      $r_{k,k+2} := 0$
      $r_{k+2,k+2} := 1$
      rot$(r_{k,k+2}, r_{k+2,k+2}, \bar{Q}_{k5}, \bar{Q}_{k6})$
    end
  end
end

这个接口程序应该执行下列任务:

1. 检查是否提供输入和输出参数的正确数目.
2. 确定输入矩阵的维数符合它们的形式说明.

3. 为输出矩阵分配存储空间.
4. 调用另一子程序，执行实际的计算.

文件 mex.h 包含 MEX 文件声明和原型. 有一些辅助的子程序，可由接口程序调用. 详细情况，读者可参考 MATLAB 的外部接口指南.

算法 9.7 算法 9.6 的 MEX 文件

```
#include "mex.h"

#define max(A, B)  ((A) > (B) ? (A) : (B))
#define min(A, B)  ((A) < (B) ? (A) : (B))

#define n     prhs[0]
#define sigma prhs[1]

#define Qbar plhs[0]
#define R    prhs[1]

void mexFunction (int nlhs, mxArray *plhs[],
                  int nrhs, mxArray *prhs[])
{ int size, nnz;

  if (nrhs != 2) {
    mexErrMsgTxt ("spqr requires two input arguments.");
  } else if (nlhs != 2) {
    mexErrMsgTxt ("spqr requires two output arguments.");
  }

  if ((mxGetM (n) != 1) || (mxGetN (n) != 1) ||
      (*mxGetPr (n) < 3.0)) {
    mexErrMsgTxt ("n must be a scalar greater or equal 3.");
  }

  if ((mxGetM (sigma) != 1) || (mxGetN (sigma) != 1)) {
    mexErrMsgTxt ("sigma must be a scalar.");
  }

  size = (int) *mxGetPr (n);
  Qbar = mxCreatDoubleMatrix (size-2, 6, mxREAL);
  if (size == 3) {nnz = 1;} else {nnz = 3*size - 9;}
  R = mxCreateSparse (size-2, size-2, nnz, mxREAL);
```

```
  QR (size, *mxGetPr (sigma), mxGetPr (Qbar), R);
}
```

算法 9.8 在 C 中计算 QR 分解 (9.48)

```
void QR (int n, double sigma, double *Qbar, mxArray *R)
{ int    i, j, k, nnz, n2, n3, n4, *ir, *jc;
  double co, *pr, si, t, tmp;

  nnz = mxGetNzmax (R); n2 = n-2;  n3 = n-3;  n4 = n-4;
  ir = mxGetIr (R);  jc = mxGetJc (R); pr = mxGetPr (R);
  /* diagonal of R */
  ir [0] = 0;  for (i = 1; i < n2; i++) {ir [3*i - 1] = i;}
  /* first upper off-diagonal of R */
  for (i = 0; i < n3; i++) {ir [3*i + 1] = i;}
  /* second upper off-diagonal of R */
  for (i = 0; i < n4; i++) {ir [3*i + 3] = i;}
  /*columns of R */
  jc [0] = 0; jc [1] = 1;
  for (j=2; j < n2; j++) {jc [j] = 3*j -3;}
  jc [n2] = nnz;

#define r(i, j) pr [k = jc [j], k + i - ir [k]]
  r (0, 0) = 1.0;  t = -2.0;  if (n > 3) {r (1, 1) = 1.0;}
  for (j = 0; j < n2; j++) {
    tmp = sigma;
    rotg (&r (j, j), &tmp, &Qbar [j], &Qbar [n2 + j]);
    rotg (&r (j, j), &t, &Qbar [2*n2 + j], &Qbar [3*n2 + j]);
    if (j < n3) {
      r (j, j+1) = 0.0;
      rot (&r (j, j+1), &r (j+1, j+1),
           Qbar [2*n2 + j], Qbar [3*n2 + j]);
    }
    tmp = 1.0;
    rotg (&r (j, j), &tmp, &Qbar [4*n2 + j], &Qbar [5*n2 + j]);
    if (j < n3) {
      t = -2.0;
      rot (&r (j, j+1), &t, Qbar [4*n2 + j], Qbar [5*n2 + j]);
      if (j < n4) {
        r (j, j+2) = 0.0;  r (j+2, j+2) = 1.0;
        rot (&r (j, j+2), &r (j+2, j+2),
             Qbar [4*n2 + j], Qbar [5*n2 + j]);
```

```
      }
    }
  }
#undef r
}
```

算法 9.9 最小二乘光滑滤子

```
function g = lsq (f, delta)

n = length (f);
alpha = sqrt (n) * delta;
e = ones (n, 1);
A = spdiags ([e -2*e e], 0:2, n-2, n);

lambda = 0;
while 1,
  [Qbar, R] = spqr (n, sqrt (lambda));
  z = QTx (Qbar, [f; zeros(n-2, 1)]);
  z = R \ z (1:n-2);

  F = norm (A' * z)^2;
  if (F <= alpha^2), break; end;

  if (lambda > 0),
    zp = QTx (Qbar, [zeros(n, 1); -z/sqrt(lambda)]);
    zp = R \ zp (1:n-2);
    Fp = -2 * z' * (z + lambda * zp);
  else
    Fp = -2 * z' * z;
  end;

  lambdaold = lambda;
  lambda = lambda - 2 * (F / Fp) * (sqrt (F) / alpha - 1);
  if (lambda <= lambdaold), break; end;
end;
g = f - A' * z;
```

作为一个例子，我们给出 C 语言的算法 9.7, 它作为算法 9.6 的 QR 分解的一个接口. 算法 9.8 是由算法 9.6 的伪代码转换成 C 语言得到. 我们提醒读者，所有的机器可读形式的程序都是可行的 (参见前言). 在 MATLAB 中，我们能执行语句

```
>> [Qbar, R] = spqr (n, sqrt(lambda));
```

计算稀疏的 QR 分解 (9.48).

以同样的方法，用 MEX 文件实现算法 9.4 和 9.5. 在 MATLAB 中，分别用语句 **y = Qx (Qbar, x)** 和 **y = QTx (Qbar, x)** 调用.

9.3.5 结论

现在我们提出算法 9.9 实现最小二乘滤子. 方程 (9.19) 的矩阵 A 构成一个稀疏矩阵.

在此方法中，只存储 A 的非零元素. 用函数 **spdiags** 实现它，此函数通过对角线性质定义了一个稀疏矩阵.

计算 QR 分解 (9.48) 是通过调用 **spqr**, 它是一个 MEX 文件，用来实现算法 9.6. 类似地，函数 **QTx** 的调用是执行算法 9.5 对应的 MEX 文件.

如果算法 9.9 应用到图 9.2 所示的测试数据，则由图 9.7 可得光滑曲线. 我们令 $\delta = 0.1$, 它是函数 $f(x)$ 中噪声的标准差.

图 9.7 用最小二乘滤子光滑图 9.2 的数据

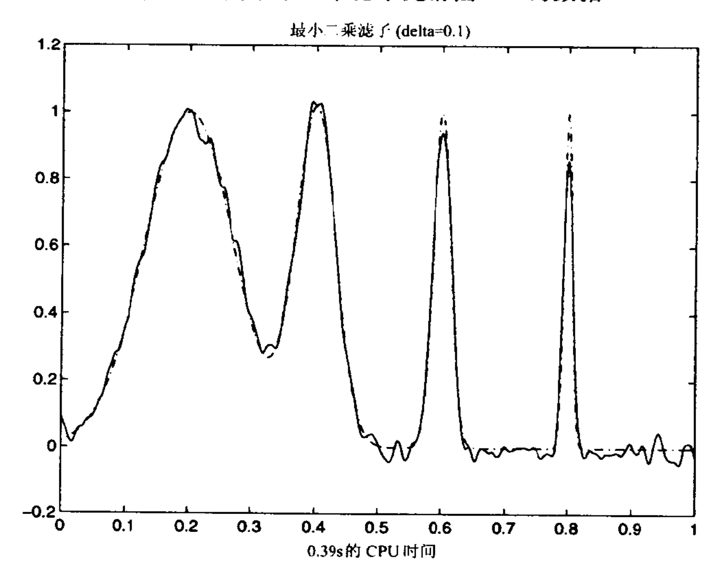


表 9.1 对应于不同数据的 CPU 时间

n	Savitzky-Golay 滤子	最小二乘滤子
10^3	0.006 sec	0.39 sec
10^4	0.036 sec	4.55 sec
10^5	0.344 sec	77.93 sec

从视觉的角度，最小二乘滤子返回一个比 Savitzky-Golay 滤子更光滑的结果， Savitzky-Golay 滤子的输出由图 9.4 给出. 当点数 n 增加时，结果甚至更好. 另一方面，执行算法 9.9 所需的 CPU 时间，本质上比执行算法 9.2 Savitzky-Golay 滤子的时间更多. 对许多不同的数据 n, 将这些 CPU 时间总结成表 9.1.

总之，最小二乘滤子能产生一个比 Savitzky-Golay 滤子更光滑的曲线 $g(x)$. 然而，为此需要编写更复杂的算法和花费更多的 CPU 时间. 最小二乘滤子的优点是只有一个参数，噪声的标准差 δ 需要估计. 这个量与实际数据的关系，比 Savitzky-Golay 滤子中的参数 n_L, n_R 和 M 更直接.

参考文献

[1] M. U. A. BROMBA AND H. ZIEGLER, *Application Hints for Savitzky-Golay Digital Smoothing Filters*, Analytical Chemistry, 53(1981), pp 1583-1586.

[2] J. J. DONGARRA, C. B. MOLER, J. R. BUNCH AND G. W. STEWART, *LINPACK Users' Guide*, SIAM Publications, Philadelphia, 1979.

[3] W. GANDER, *On the Linear Least Squares Problem with a Quadratic Constraint*, Habilitationsschrift ETH Zürich, STAN-CS-78-697, Stanford University, 1978.

[4] W. GANDER, *Least Squares with a Quadratic Constraint*, Numer. Math. , 36(1981), pp. 291-307.

[5] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Second Edition, The Johns Hopkins University Press, Baltimore, 1989.

[6] C. L. LAWSON, R. J. HANSON, D. R. KINCAID AND F. T. KROGH, *Basic Linear Algebra Subprograms for Fortran Usage*, ACM Trans Math Softw, 5(1979), pp. 308-325.

[7] W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY AND W. T. VETTERLING, *Numerical Recipes*, Cambridge University Press, Cambridge, 1986.

[8] W. H. PRESS AND S. A. TEUKOLSKY, *Savitzky-Golay Smoothing Filters*, Computers in Physics, 4(1990), pp. 669-672.

[9] C. H. REINSCH, *Smoothing by Spline Functions. II*, Numer. Math., 16(1971), pp. 451-454.

[10] A. SAVITZKY AND M. J. E. GOLAY, *Smoothing and Differentiation of Data by Simplified Least Squares Procedures*, Analytical Chemistry, 36(1964), pp. 1627-1639.

[11] J. STEINIER, Y. TERMONIA AND J. DELTOUR, *Comments on Smoothing and Differentiation of Data by Simplified Least Square Procedure*, Analytical Chemistry, 44(1972), pp. 1906-1909.

[12] U. VON MATT, *Large Constrained Quadratic Problems*, Verlag der Fachvereine, Zürich, 1993.

[13] H. ZIEGLER, *Properties of Digital Smoothing Polynomial (DISPO) Filters*, Applied Spectroscopy, 35(1981), pp. 88-92.

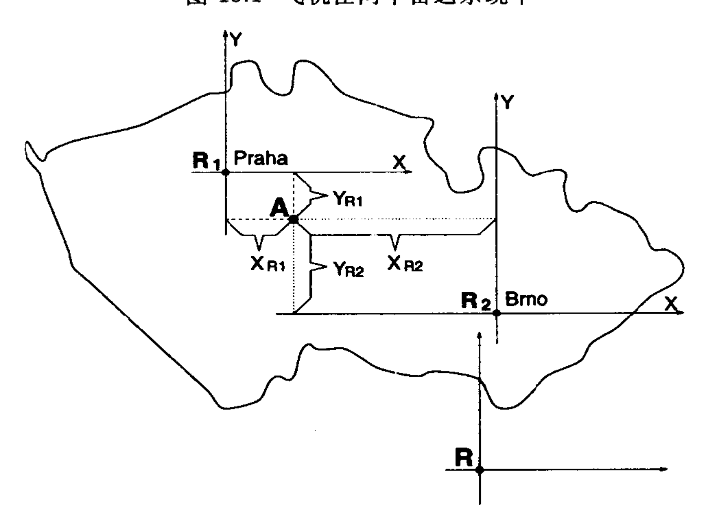
第十章　雷达问题

S. Barton̆ and I. Daler

10.1　引言

在空中交通远程控制中心，多雷达监视系统接收到来自全球各个雷达的不同信息．来自每个雷达的信息除了其它信息之外，还包含有该雷达看到的飞机的坐标 x, y 和 z(指相对于该雷达的直角坐标系而言)．两个雷达 R_1, R_2 的二维图形的例子画在图 10.1 中．可以看到，飞机是由雷达 R_1 的坐标 $[x_{R_1}, y_{R_1}]$ 和雷达 R_2 的坐标 $[x_{R_2}, y_{R_2}]$ 来描述的．对多雷达监视系统而言，必须以“绝对”坐标系 (以 R 为原点) 来工作．所有雷达的有关飞机 A 的数据都要变换到这个坐标系．这意味着在这个坐标系中，飞机 A 可以用几种观点来描述，它们分别有自己相应的标准．(例如，飞机可以用它最可能在的位置点来描述)．在以 R 为原点的坐标系中，远程控制系统可以从监视器的视觉信息追踪飞机的相关位置．利用选定的窗口，多雷达监视系统的控制程序可以监视飞机活动的整个区域．为了简单，让 R 为原点的绝对坐标系和雷达 R_2 的坐标系相重合．对多雷达监视系统而言，必须求解以下基本问题：

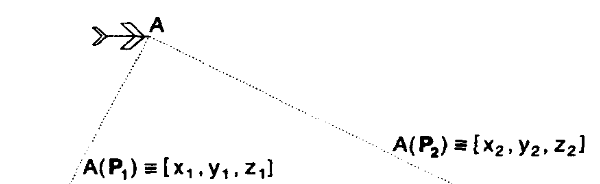
图 10.1　飞机在两个雷达系统中



让雷达 R_1 和 R_2 位于地球表面的 P_1 和 P_2．飞机 A 在雷达 R_1 的当地直角坐标系 $\mathbf{P_1}$ 中被“看到”的坐标是 $[x_1, y_2, z_1] \equiv A(\mathbf{P_1})$，必须求出它在雷达 R_2 的直角坐标系 $\mathbf{P_2}$ 中的坐标 $[x_2, y_2, z_2] \equiv A(\mathbf{P_2})$(见图 10.2)．地球表面的点 P_1, P_2 的位置由地理坐标 $[f_1, l_1]$ 和 $[f_2, l_2]$ 给定，f_1, f_2 是地理纬度，l_1, l_2 是地理经度．地理坐标 $\mathbf{G} \equiv [f, l]$ 常用度，分和秒表示 $(-90° \le f \le 90°, -180° \le l \le 180°)$．让直角坐标系统 $(P_1; +x_1, +y_1, +z_1) \equiv \mathbf{P_1}, (P_2; +x_2, +y_2, +z_2) \equiv \mathbf{P_2}$ 的原点分别在 P_1, P_2(参看图 10.5)．x_1y_1 平面是在椭球体的点 P_1 的切平面，x_1 轴位于纬度 f_1 的平面内，y_1 轴位于子午线 l_1

的平面内. z_1 轴正向指向地心. x_1 和 y_1 轴的正向分别指向经度和纬度增加的方向. 坐标系 $\mathbf{P_2}$ 各轴的方向相同.

图 10.2 飞机在两个雷达系统中



10.2 将角度化成弧度

地理坐标 $[f, l]$ 常用度分秒以及它们的小数表示. 为了以后的计算, 必须将角度 f, l 化为弧度. 由于我们会多次用到它们, 所以设计了一个 MATLAB 函数 deg2rad($\alpha°$, [′], [″]). 这些函数的输入变量 α 是含 1 至 4 个分量的矢量 (参见算法 10.1, 表 10.1 和例 1).

算法 10.1 函数 deg2rad

```
function rad = deg2rad(alpha)
%----------------------------
% conversion: alpha deg., min., sec.  ->  radians
%
d = length(alpha);
if d > 4
  disp(alpha), error('invalid input list in function deg2rad')
end
alpha = [alpha(:); zeros(4-d,1)];
alpha(3) = alpha(3) + alpha(4)/100;
rad = pi/180*((alpha(3)/60 + alpha(2))/60 + alpha(1));
```

表 10.1 角度 α 的组成

分量的个数	输入角 α
1	整数. 十分数 °
2	整数 ° 整数. 十分数 ′
3	整数 ° 整数 ′ 整数. 十分数 ″
4	整数 ° 整数 ′ 整数 ″ 整数 (十分数)″

例 1

我们验证一下如表 10.1 所示的输入角 α 的所有四种可能性. 对给定的输入矢量, 总是得到相同的弧度值.

```
>> format long
>> a = [16.641038888888888];        % 1 component
>> b = [16 38.462333333333333];     % 2 components
>> c = [16 38 27.74];               % 3 components
>> d = [16 38 27 74];               % 4 components
>> radians = [deg2rad(a) deg2rad(b) deg2rad(c) deg2rad(d)]

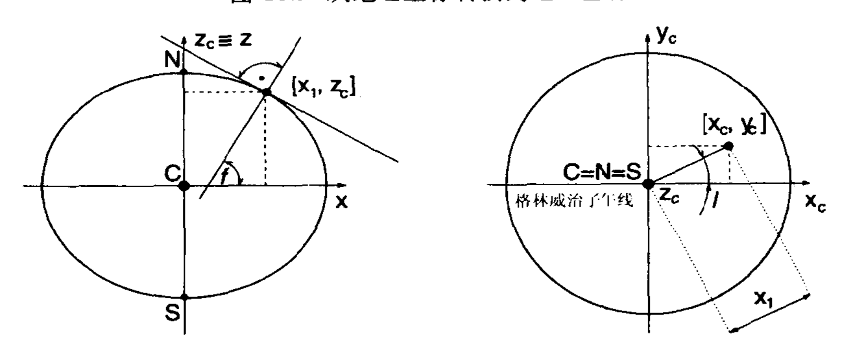
radians =

0.29044091956353 0.29044091956353 0.29044091956353 029044091956353
```

10.3 地理坐标化为地心坐标

在本节引入一个函数来把 P 点的地理坐标 $[f,l]$ 变换成地心直角坐标：$(C;+x_c,+y_c,+z_c)\equiv \mathbf{C}$，它的原点 C 在地球中心. z_c 轴沿地球的极轴，正向指向北极. x_c 轴通过 0 度和 180 度的子午线，正向指向 0 度子午线. y_c 轴垂直于 x_c 和 z_c 轴，正方向指向 90 度的子午线 (见图 10.5). 为了使地理坐标 $[f,l]$ 变换到地心坐标 $[x_c,z_c,y_c]$(按照图 10.3)，我们编写了一个 MATLAB 函数 gg2gc(f,l). 地球是一个不规则球体，为了计算，我们用园椭球体对它作近似描述. 两个球体的旋转轴相同. 椭球体的横截面是 (Krasovsky) 椭园，长半轴是 $A=6378.245[km]$，短半轴是 $B=6356.863[km]$. 为了写出变换方程，使用椭园的参数描述是很方便的 (参见图 10.4).

图 10.3 从地理坐标转换到地心坐标



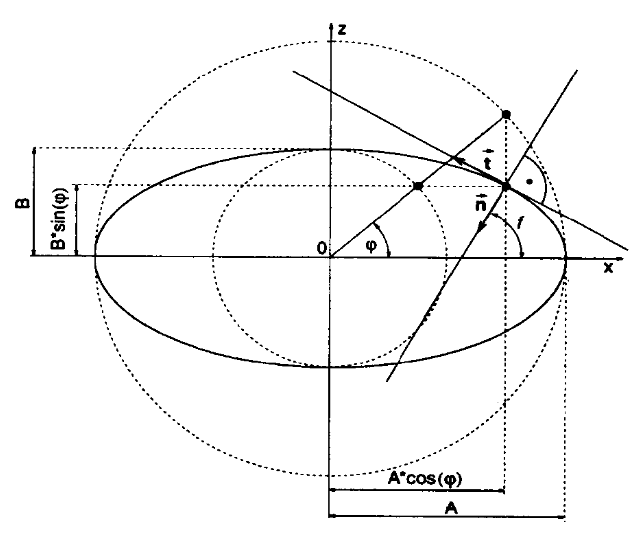
重要的是记住，地理坐标系并非是球心坐标系. 地理坐标的纬度是通过测量物体 (如太阳，月亮，已知星体) 在天空与地平线的夹角来确定，也即地球表面的切平面与该物体的夹角. 描述椭球的常用参数是

$$x(\varphi)=A\cos(\varphi),\qquad z(\varphi)=B\sin(\varphi).$$

为了把角度 f 与 x,z 相联系，先计算在 $[x,z]$ 点的法向矢量 $\vec{n}$,

$$\vec{t}\equiv\begin{pmatrix}\dot{x}\\ \dot{z}\end{pmatrix}=\begin{pmatrix}-A\sin(\varphi)\\ B\cos(\varphi)\end{pmatrix}\quad\Longrightarrow\quad\vec{n}\equiv\begin{pmatrix}-\dot{z}\\ \dot{x}\end{pmatrix}=\begin{pmatrix}-B\cos(\varphi)\\ -A\sin(\varphi)\end{pmatrix}.$$

图 10.4 用法向矢量的斜率描述椭圆



所以有

$$\tan(f) = -\frac{\dot{x}}{\dot{z}} = \frac{A}{B}\tan(\varphi).$$

我们要的是 $\sin(\varphi)$ 和 $\cos(\varphi)$. 从 $\tan(\varphi) = \frac{B}{A}\tan(f)$ 得到:

$$\cos(\varphi) = \frac{1}{\sqrt{1+\tan^2(\varphi)}}, \qquad \sin(\varphi) = \tan(\varphi)\cos(\varphi).$$

下面给出对应的 MATLAB 函数作为算法 10.2.

算法 10.2 函数 gg2gc

```
function [P] = gg2gc(f, l)
% transformation geographic. -> geocentric. coordinates
% f, l in radians
%
A = 6378.245; B = 6356.863;   % Krasovsky ellipse
tanfi = B/A*tan(f);
cosfi = 1/sqrt(1 + tanfi^2);
sinfi = tanfi*cosfi;
P = [ A*cosfi*cos(l);  A*cosfi*sin(l);  B*sinfi ];
```

例 2

通过计算 Brno 的球心坐标可以核对两个函数.

```
>> f = deg2rad([49 3 37 2]);
>> l = deg2rad([16 38 27 74]);
>> BRNO = gg2gc(f,l)
```

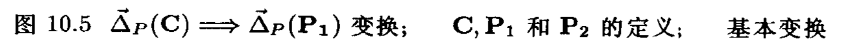
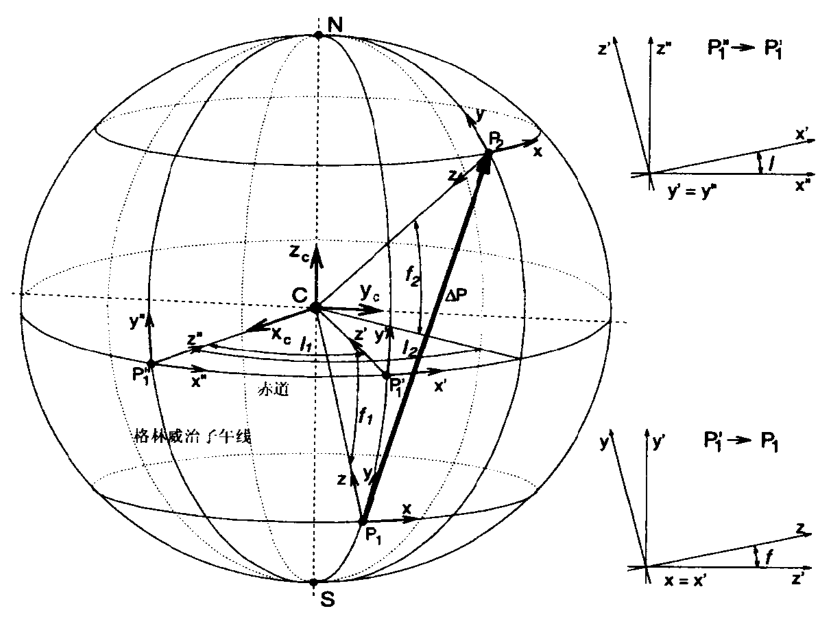


图 10.5 $\vec{\Delta}_P(\mathbf{C}) \Longrightarrow \vec{\Delta}_P(\mathbf{P_1})$ 变换； $\mathbf{C}, \mathbf{P}_1$ 和 $\mathbf{P_2}$ 的定义； 基本变换




```
BRNO =

  1.0e+003 *
  4.01206298678276
  1.19917767600837
  4.79503887137397
```

10.4 变换

现在进行坐标系 $\mathbf{P_1}$ 到坐标系 $\mathbf{P_2}$ 的变换．要更详细地了解坐标变换，请看文献 [1,2]．这个复杂的变换包含四个局部变换，具体来说，就是一个平移和三个转动．由于坐标系 $\mathbf{P_1}$ 在每次变换中都发生改变，我们用以下足标来区分它们：

$$\mathbf{P_1^i}, \quad i = 0, \cdots, 4, \quad \text{这里 } \mathbf{P_1^0} \equiv \mathbf{P_1}, \quad \mathbf{P_1^4} \equiv \mathbf{P_2}.$$

1. 平移变换

平移量的值 $[\Delta x, \Delta y, \Delta z] = P_2(\mathbf{P_1}) - P_1(\mathbf{P_1}) \equiv \vec{\Delta}_P$ 是点 P_2 在第一个坐标系的坐标，我们把它标为 $P_2(\mathbf{P_1})$．在坐标系 $\mathbf{C}$ 中， $\vec{\Delta}_P$ 是已知的： $\vec{\Delta}_P(\mathbf{C}) = P_2(\mathbf{C}) - P_1(\mathbf{C})$．必须把 $\vec{\Delta}_P(\mathbf{C})$ 变换成 $\vec{\Delta}_P(\mathbf{P_1})$．这次变换如图 10.5 所示，可按如下步骤来计算：

1. 把坐标系 $\mathbf{P_1''}$ 的原点 P_1'' 放在赤道与格林威治子午线的交点上，从图 10.5 看到，这个变换是

十分简单的.

$$\begin{pmatrix}\Delta x\\ \Delta y\\ \Delta z\end{pmatrix}_{\mathbf{P}_1''}=\begin{pmatrix}\Delta y\\ \Delta z\\ -\Delta x\end{pmatrix}_{\mathbf{C}}=M\times\begin{pmatrix}P_{2x}-P_{1x}\\ P_{2y}-P_{1y}\\ P_{2z}-P_{1z}\end{pmatrix}_{\mathbf{C}},\ \text{这里}\ M=\begin{pmatrix}0&1&0\\0&0&1\\-1&0&0\end{pmatrix}.$$

2. 为了移动 $\mathbf{P}_1''$ 到 $\mathbf{P}_1'$, 我们绕 y'' 轴旋转角度 l_1. 这个变换用旋转矩阵 R_1 来描述:

$$\begin{pmatrix}\Delta x\\ \Delta y\\ \Delta z\end{pmatrix}_{\mathbf{P}_1'}=R_1\times\begin{pmatrix}\Delta x\\ \Delta y\\ \Delta z\end{pmatrix}_{\mathbf{P}_1''},\ \text{这里}\ R_1=\begin{pmatrix}\cos(l_1)&0&\sin(l_1)\\0&1&0\\-\sin(l_1)&0&\cos(l_1)\end{pmatrix}.$$

3. 为了移动 $\mathbf{P}_1'$ 到 P_1, 我们绕 x' 轴旋转角度 f_1. 这个变换用旋转矩阵 R_2 来描述:

$$\begin{pmatrix}\Delta x\\ \Delta y\\ \Delta z\end{pmatrix}_{\mathbf{P}_1}=R_2\times\begin{pmatrix}\Delta x\\ \Delta y\\ \Delta z\end{pmatrix}_{\mathbf{P}_1'},\ \text{这里}\ R_2\equiv\begin{pmatrix}1&0&0\\0&\cos(f_1)&\sin(f_1)\\0&-\sin(f_1)&\cos(f_1)\end{pmatrix}.$$

4. 现在坐标系 $\mathbf{P_1}$ 处在一个任意的位置. 总结步骤 1–4, 平移矢量 $\vec{\Delta}_P(\mathbf{P_1})$ 可以表述为

$$\begin{pmatrix}\Delta x\\ \Delta y\\ \Delta z\end{pmatrix}_{\mathbf{P}_1}=R_2\times R_1\times M\times\begin{pmatrix}P_{2x}-P_{1x}\\ P_{2y}-P_{1y}\\ P_{2z}-P_{1z}\end{pmatrix}_{\mathbf{C}}.$$

使用关系式

$$A(\mathbf{P}_1^1)\equiv\begin{pmatrix}x\\ y\\ z\end{pmatrix}_{\mathbf{P}_1^1}=A(\mathbf{P}_1)-\begin{pmatrix}\Delta x\\ \Delta y\\ \Delta z\end{pmatrix}_{\mathbf{P}_1},$$

我们从飞机先前的坐标 $A(\mathbf{P_1})$ 推出了新的坐标 $A(\mathbf{P}_1^1)$.

2. 第一次旋转

将坐标系 $\mathbf{P}_1^1$ 绕 x 轴旋转角度 f_1 以使 y 轴平行于地球的极轴 z_c. 然后使用下式

$$A(\mathbf{P}_1^2)\equiv\begin{pmatrix}x\\ y\\ z\end{pmatrix}_{\mathbf{P}_1^2}=R_3\times\begin{pmatrix}x\\ y\\ z\end{pmatrix}_{\mathbf{P}_1^1},\ \text{这里}\ R_3=R_2^T\equiv\begin{pmatrix}1&0&0\\0&\cos(f_1)&-\sin(f_1)\\0&\sin(f_1)&\cos(f_1)\end{pmatrix},$$

可从飞机原来的坐标 $A(\mathbf{P}_1^1)$ 求出新坐标 $A(\mathbf{P}_1^2)$.

3. 第二次旋转

将坐标系 $\mathbf{P}_1^2$ 绕 y 轴旋转角度 $\Delta l=l_2-l_1$ 以使 $\mathbf{P}_1^2$ 和 $\mathbf{P_2}$ 的两个 x 轴重合. 然后使用下式

$$A(\mathbf{P}_1^3)\equiv\begin{pmatrix}x\\ y\\ z\end{pmatrix}_{\mathbf{P}_1^3}=R_4\times\begin{pmatrix}x\\ y\\ z\end{pmatrix}_{\mathbf{P}_1^2},\ \text{这里}\ R_4\equiv\begin{pmatrix}\cos(\Delta l)&0&\sin(\Delta l)\\0&1&0\\-\sin(\Delta l)&0&\cos(\Delta l)\end{pmatrix},$$

可从飞机原来的坐标 $A(\mathbf{P}_1^2)$ 求出新坐标 $A(\mathbf{P}_1^3)$.

4. 第三次旋转

将坐标系 $\mathbf{P}_1^3$ 绕 x 轴旋转角度 f_2 以使它和坐标系 $\mathbf{P_2}$ 重合. 然后使用下式

$$A(\mathbf{P_2}) \equiv \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\mathbf{P}_1^4} = R_5 \times \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\mathbf{P}_1^3}, \text{这里 } R_5 \equiv \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(f_2) & \sin(f_2) \\ 0 & -\sin(f_2) & \cos(f_2) \end{pmatrix}.$$

可从飞机先前的坐标 $A(\mathbf{P}_1^3)$ 求出最后的坐标 $A(\mathbf{P_2}) \equiv A(\mathbf{P}_1^4)$.

10.5 最终的算法

以上步骤可以合并起来. 使用矩阵 $R_1 - R_5$, 得到全部的变换为

$$A(\mathbf{P_2}) \equiv \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\mathbf{P_2}} = R_5 \times R_4 \times R_2^{\mathrm{T}} \times \left(A(\mathbf{P_1}) - R_2 \times R_1 \times M \times \begin{pmatrix} P_{2x} - P_{1x} \\ P_{2y} - P_{1y} \\ P_{2z} - P_{1z} \end{pmatrix}_{\mathbf{C}} \right).$$

这个算法可以作为 MATLAB 函数 `transf` 的基础 (参见算法 10.3).

$$A(\mathbf{P_2}) = \mathtt{transf}(f_1, l_1, A(\mathbf{P_1}), f_2, l_2).$$

10.6 实际例子

我们的问题已经完全解决. 可以用实际例子来检验它.

```
>> format long
>> f1 = deg2rad([49 45 18]);              % Praha
>> l1 = deg2rad([14 7 58]);               %   position
>> f2 = deg2rad([49 3 37 2]);             % Brno
>> l2 = deg2rad([16 38 27 74]);           %   position
>> A1praha = [200; 140; -5.790];          % 1. A. seen from Praha
>> A2praha = [157; 397; -9.870];          % 2. A. seen from Praha
>> A1brno = transf(f1, l1, A1praha, f2, l2)% 1. A. seen from Brno
A1brno =
   1.0e+02 *
   0.23632088742730
   2.13625791235094
  -0.06831714145899
>> a1praha = transf(f2, l2, A1brno, f1, l1)% inv. transformation
a1praha =
   1.0e+02 *
   2.00000000000000
   1.40000000000000
```

```
  -0.05790000000000
>> A2brno = transf(f1, l1, A2praha, f2, l2)% 2. A. seen from Brno
A2brno =
   1.0e+02 *
  -0.10857206384904
   4.71937837075423
  -0.06683743260798
>> a2praha = transf(f2, l2, A2brno, f1, l1)% inv. transformation
a2praha =
   1.0e+02 *
   1.57000000000000
   3.97000000000000
  -0.09870000000000
```

算法 10.3 函数 transf

```
function [A2] = transf(f1, l1, A1, f2, l2)
%-----------------------------------------
% the airplane radar position recalculation
% f, l = geographic radar positions in radians
% A1 = [x;  y; z](P1) = position of airplane seen by radar at P1
%
M  = [   0       1        0
         0       0        1        % M = almost permutations
        -1       0        0    ];  % matrix
%
R1 = [ cos(l1)   0      sin(l1)
         0       1        0        % R1 = Displacement
     -sin(l1)    0      cos(l1)];  %       1. rotation matrix
%
R2 = [  1        0        0
        0      cos(f1)  sin(f1)    % R2 = Displacement
        0     -sin(f1)  cos(f1)];  %       2. rotation matrix
%
P1c = gg2gc(f1, l1);               % 1. radar    geocentric
P2c = gg2gc(f2, l2);               % 2. radar   coordinates
%
DPc = P2c - P1c;                   % DPc - radar's displacement
%                                    (1. radar pos. [f=0, l=0])
T = R2*R1*M*DPc;                   % - radar's displacement
%                                    (1. radar pos. [f1, l1])
a = A1 - T;                        % A. pos. after translation
```

```
%
Dl = l2 - l1;                       % longitude difference
%
R4 = [ cos(Dl)   0      sin(Dl)
         0       1        0         % R4 = rotation matrix,
      -sin(Dl)   0      cos(Dl)];   %      rot. around y by l2-l1
%
R5 = [   1       0        0
         0     cos(f2)  sin(f2)     % R5 = rotation matrix,
         0    -sin(f2)  cos(f2)];   %      rot.  around x by f2
%
A2 = R5*R4*R2'*a;                   % A2 = [x; y; z](P2)
%                                     position of airplane
%                                     seen by radar at P2
% end of function
```

感谢

作者感谢 J.Hřebíček 和 W.Gander 对改进本章所提的建议和帮助.

参考文献

[1] H. J. Bartsch, *Taschenbuch Mathematischer Formeln*, Fachbuchverlag, Leipzig, 1991.

[2] D. G. Zill and M. R. Cullen, *Advanced Engineering Mathematics*, PWSKENT, Boston, 1992.

第十一章　圆的保形映射

H.J. Halin, L. Jaschke

11.1　引言

映射是数学方法，它常被用于求解在不规则形状体内的流体问题. 自从出现超级计算机以来，这种技术在生成数值网格方面已经变得非常重要 [1]. 在流体力学的入门课程中，学生学习了如何计算不可压缩流体绕 "Joukowski 机翼" 的环流，它表示最简单的飞机机翼形状. 绕机翼流体的物理平面位于复平面 $p = u + \mathrm{i}v$ 内，其中 $\mathrm{i} = \sqrt{-1}$. Joukowski 变换的优点是给出了一个从 $z = x + \mathrm{i}y$ 平面到 p 平面的保形映射，它把绕机翼流体的计算转变为绕圆柱流体的非常简单的计算. Joukowski 变换的映射函数 $p = f(z) = u(z) + \mathrm{i}v(z)$ 的一个特殊形式是

$$p = \frac{1}{2}(z + \frac{a^2}{z}). \tag{11.1}$$

在这一章我们将展示如何借助符号计算语言和计算机代数处理在应用映射方法时所需的数学变换. 为说明有关的基本步骤，与其选择一个大的物理问题，那可能超出本书的范围，倒不如选择一个保形映射的非常简单的应用.

11.2　问题概要

将 x 和 y 表成某参数 t 的函数，从而使得 $z(t) = x(t) + \mathrm{i}y(t)$. 由此得到 $p(z(t)) = f(z(t))$ 成立. 将 $z(t)$ 的表达式代入 $p(z) = f(z(t))$, 得到

$$P(t) = U(t) + \mathrm{i}V(t) = p(z(t)) \tag{11.2}$$

在等号两边关于 t 求导，得

$$\dot{P}(t) = p'(z(t))\dot{z}(t), \tag{11.3}$$

和

$$\ddot{P}(t) = p''(z(t))(\dot{z}(t))^2 + p'(z(t))\ddot{z}(t) \tag{11.4}$$

其中点 ˙ 和分号 ′ 分别表示对 t 和 z 求导.

至此所列出的关系式对处理下面的问题是必要的. 我们假设映射函数 p 是 [2] 中二阶常微分方程

$$z^2p'' + zp' + (\alpha z^2 + \beta z + \gamma)p = 0 \tag{11.5}$$

满足初始条件 $p'(0) = p'_0$ 和 $p(0) = p_0$ 的解.

显然映射 $z(t)$ 到区域 $P(t) = U(t) + iV(t)$ 上的问题需要解二阶常微分复方程 (11.5). 这可以由下列过程解决：对所考虑的问题取 $z(t)$ 是半径为 r 的圆，即 $z(t) = re^{\mathrm{i}t}$. 因此 $z(t)$ 关于 t 的导数可以轻易地得到. 将 $z(t)$, $\dot{z}(t)$ 和 $\ddot{z}(t)$ 分别代入方程 (11.2) -(11.5), 便可以在这些方程中消去 z 和它对 t 的导数. 于是在 (11.3) 和 (11.4) 中，p' 和 p'' 可以用 $\dot{P}(t)$ 和 $\ddot{P}(t)$ 表示. 最后利用这些表达式

和 (11.2), 二阶常微分方程变成仅仅是关于 $P(t)$, $\dot{P}(t)$ 和 $\ddot{P}(t)$ 的方程. 考虑到 (11.2), 我们可按实部和虚部化简这个方程. 这样产生了两个耦和的二阶微分方程组. 通过标准的数值积分可求得解 $U(t)$ 和 $V(t)$.

对下面两组初始条件和参数 r 可以求得相应的解:

$$\begin{array}{lcllcl} r & = & 1 & r & = & 0.6 \\ U(0) & = & -0.563 & U(0) & = & -0.944 \\ \dot{U}(t) & = & 0 & \dot{U}(t) & = & 0 \\ V(0) & = & 0 & V(0) & = & 0 \\ \dot{V}(t) & = & 0.869 & \dot{V}(t) & = & 0.658 \end{array}$$

在这两个运算过程中, 我们在区间 $(0 \le t \le 6\pi)$ 上积分. 而且, 使用了参数值 $\alpha = 1, \beta = 0.5$ 和 $\gamma = -4/9$. 下面这后三个参数值将被记为 a, b 和 c.

11.3 MAPLE 解

上面的解题步骤并不难执行. 但是因为存在一些需要灵活处理的问题, 手工推导公式和用高级语言编程都容易出错, 并且花费时间. 下面将证实象 MAPLE 这样的高级数学计算工具, 凭借它大量的符号, 数字和图像功能能使求解的全过程或至少部分过程更严密和精巧.

处理例题的第一个步骤是分别导出关于 $U(t)$ 和 $V(t)$ 的两个二阶常微分方程. 我们从定义复平面 $P = U + \mathrm{i}V$ 上变形的圆 $P(t) = p(z(t))$ 的参数表达式开始.

```
> p(z(t))=P(t):
```

现在我们要表示未知函数 $p(z(t))$ 关于 t 的一阶导数. 为此我们应用 MAPLE 的求导算子 `diff` 并求关于 z 的一阶导数, 即 $p'(t) = D(p)(z(t))$, 记为 `pprime`.

```
> pprime:=solve(diff(",t),D(p)(z(t)));
```

$$pprime := \frac{\frac{\partial}{\partial t}\,\mathrm{P}(t)}{\frac{\partial}{\partial t}\,\mathrm{z}(t)}$$

类似地, 通过对 p 两次运用算子 D, 我们得到一个求解 $p''(t)$ 的表达式, 我们将它记为 `p2prime`.

```
> p2prime:=solve(diff("",t,t),D(D(p))(z(t)));
```

$$p2prime := -\frac{\mathrm{D}(p)(\mathrm{z}(t))\,(\frac{\partial^2}{\partial t^2}\,\mathrm{z}(t)) - (\frac{\partial^2}{\partial t^2}\,\mathrm{P}(t))}{(\frac{\partial}{\partial t}\,\mathrm{z}(t))^2}$$

因为按定义 $D(p)(z(t))$ 等于 $p'(t)$, 我们可以将这个替换用在 `p2prime` 的表达式中.

```
> D(p)(z(t)):=pprime;
```

$$\mathrm{D}(p)(\mathrm{z}(t)) := \frac{\frac{\partial}{\partial t}\,\mathrm{P}(t)}{\frac{\partial}{\partial t}\,\mathrm{z}(t)}$$

p2prime 的新形式是

```
> p2prime;
```

$$-\frac{\frac{(\frac{\partial}{\partial t}\mathrm{P}(t))(\frac{\partial^2}{\partial t^2}\mathrm{z}(t))}{\frac{\partial}{\partial t}\mathrm{z}(t)}-(\frac{\partial^2}{\partial t^2}\mathrm{P}(t))}{(\frac{\partial}{\partial t}\mathrm{z}(t))^2}$$

现在可以代入复方程 (11.5), 方程将被命名为 ode.

```
> ode:=z(t)^2*p2prime+z(t)*pprime+(a*z(t)^2+b*z(t)+c)*P(t);
```

$$ode := -\frac{\mathrm{z}(t)^2\left(\frac{(\frac{\partial}{\partial t}\mathrm{P}(t))(\frac{\partial^2}{\partial t^2}\mathrm{z}(t))}{\frac{\partial}{\partial t}\mathrm{z}(t)}-(\frac{\partial^2}{\partial t^2}\mathrm{P}(t))\right)}{(\frac{\partial}{\partial t}\mathrm{z}(t))^2}+\frac{\mathrm{z}(t)(\frac{\partial}{\partial t}\mathrm{P}(t))}{\frac{\partial}{\partial t}\mathrm{z}(t)}+(a\mathrm{z}(t)^2+b\mathrm{z}(t)+c)\mathrm{P}(t) \qquad (11.6)$$

至此，除了注意到 $z(t)$ 是个圆的事实，处理问题的过程已被严格地公式化．下一步这个信息将被提供.

```
> z(t):=r*exp(I*t):
```

这样我们可以确定 $P(t)$ 为一个仅依赖 t 的函数.

```
> P(t):=U(t)+I*V(t):
```

我们的复方程 ode 将被分成两个耦和的有相同阶数的方程 re 和 im. 所有实分量将被存于 re. 同样所有虚分量组成 im. 在每部分中我们都用圆的三角函数表达式代替它的复指数函数．这将由 MAPLE 中的变量 trig 和组合命令 combine(...,trig) 实现．执行这个分解的命令具体为

```
> re:=combine(evalc(Re(ode)),trig);
```

$$\begin{aligned} re \;&:=\; -(\frac{\partial^2}{\partial t^2}\mathrm{U}(t))+\mathrm{U}(t)\,a\,r^2\cos(2\,t)+\mathrm{U}(t)\,b\,r\cos(t)+\mathrm{U}(t)\,c \\ &-\; r^2\,\mathrm{V}(t)\,a\sin(2\,t)-r\,\mathrm{V}(t)\,b\sin(t) \end{aligned}$$

```
> im:=combine(evalc(Im(ode)),trig);
```

$$\begin{aligned} im \;&:=\; -(\frac{\partial^2}{\partial t^2}\mathrm{V}(t))+r^2\,\mathrm{U}(t)\,a\sin(2\,t)+r\,\mathrm{U}(t)\,b\sin(t)+\mathrm{V}(t)\,a\,r^2\cos(2\,t) \\ &+\; \mathrm{V}(t)\,b\,r\cos(t)+\mathrm{V}(t)\,c \end{aligned}$$

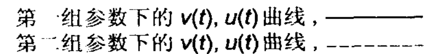
由上面几行可见这些项还没有很好地组合．因此我们将 $U(t)$ 和 $V(t)$ 的系数分别集中于 re 和 im. 这样产生了

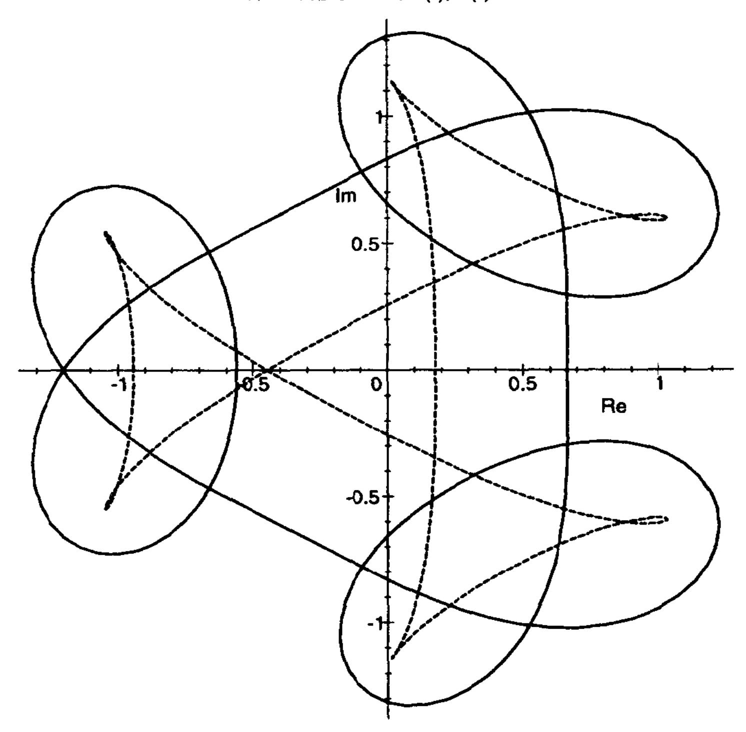
```
> re:=collect(collect(re,U(t)),V(t))=0:
> im:=collect(collect(im,U(t)),V(t))=0:
```

为了积分我们将组合关于 $P(t)$ 的实部和虚部的微分方程，形成方程组 odes.

```
> odes:= re,im;
```

图 11.1





odes :=

$$(-b\,r\sin(t) - a\,r^2\sin(2\,t))\,\mathrm{V}(t) + (a\,r^2\cos(2\,t) + b\,r\cos(t) + c)\,\mathrm{U}(t) - (\frac{\partial^2}{\partial t^2}\,\mathrm{U}(t)) = 0,$$

$$(a\,r^2\cos(2\,t) + b\,r\cos(t) + c)\,\mathrm{V}(t) + (a\,r^2\sin(2\,t) + b\,r\sin(t))\,\mathrm{U}(t) - (\frac{\partial^2}{\partial t^2}\,\mathrm{V}(t)) = 0$$

对这样一组微分方程求解其数值解是合理的且会省时间，因为 MAPLE 运算的模式在编译程序过程中是交互的，而不是进行数值咀嚼．而且，现在 MAPLE 提供的只是 4-5 阶 Runge-Kutta-Fehlberg 算法．因为 MAPLE 允许输出 Fortran 和 C 的符号，所以用它导出的方程，可以自动地在 Fortran 或 C 环境中进一步应用．

MAPLE 能够借助它的 4-5 阶 Runge-Kutta-Fehlberg 算法，为我们的问题提供一个数值解．在运用这个算法之前，我们按我们的第一组参数集给定初值

```
>  init:=V(0)=0,U(0)=-.563,D(V)(0)=.869,D(U)(0)= 0:
```

带着这些初始条件 MAPLE 的程序 dsolve 给出一个解，我们将它看作一个过程 F, 它能够对变化的变量 t 提供 $U(t)$ 和 $V(t)$ 的数值．当然我们还需要给定其它参数，即 a，b，c 和 r．

```
>  a:=1:  b:=0.5:  c:=-0.444444443:  r:=1:
>  F1:=dsolve({odes, init}, {U(t),V(t)},numeric);
```

$$F1 := \textbf{proc}(rkf45_x)\ \ldots\ \textbf{end}$$

因为我们要借助图像表示解，我们必须在区间 $(0 \leq t \leq 6\pi)$ 内的例如 201 个均分点上执行过程 F1，并暂时将 $U(t)$ 和 $V(t)$ 的值存储在一个列表中.

```
> numpts:= 200:  Tend:= 6*Pi:
> L1:=[seq(subs(F1(j*Tend/numpts),[U(t),V(t)]), j=0..numpts)]:
```

其中 numpts 表示点数. L1 是列表的列表. 每次运行程序 F1 时，新的一组 $t, U(t)$ 和 $V(t)$ 将被添加到这个列表中. 最后我们用一个 MAPLE 命令产生一个 $V(t)$ 对 $U(t)$ 曲线，如图 11.1(线 ——).

```
> plot(L1);
```

用第二组初值和新参数值 r 我们将得到另一个解，它也被画在图 11.1 中（线 - - -).

```
> init:=V(0)=0,U(0)=-.944,D(V)(0)=.658,D(U)(0)= 0:
> r:=0.6:
> F2:=dsolve({odes, init}, {U(t),V(t)},numeric):
> L2:=[seq(subs(F2(j*Tend/numpts),[U(t),V(t)]), j=0..numpts)]:
```

现在图 11.1 中的曲线可由下面的命令产生:

```
> plot({L1,L2},labels=[Re,Im],linestyle=[0,3] ,color=black, font=[HELVETICA,12]);
```

11.4 MATLAB 解

在前一节中微分方程的数值积分也可以用 MATLAB 实现. 最新的 MATLAB 5.0 版本为解复常微分方程提供了几种函数.

我们从前一节方程 (11.6) 开始. 将

$$z(t) := r\mathrm{e}^{-\mathrm{i}t}$$

代入下面的微分方程

```
> ode = 0;
```

$$-(\frac{\partial^2}{\partial t^2}P(t)) + (ar^2(e^{(it)})^2 + bre^{(it)} + c)P(t) = 0 \tag{11.7}$$

为用 MATLAB 解这个常微分方程，我们将这个二阶常微分方程转变为两个一阶微分方程的方程组. 用替换

$$\mathrm{y}(t) := \begin{bmatrix} p(t) \\ \dot{p}(t) \end{bmatrix},$$

方程 (11.7) 可以被写成

$$\dot{\mathrm{y}} = \begin{bmatrix} \dot{p}(t) \\ \ddot{p}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ ar^2\mathrm{e}^{2\mathrm{i}t} + br\mathrm{e}^{\mathrm{i}t} + c & 0 \end{bmatrix} \underbrace{\begin{bmatrix} p(t) \\ \dot{p}(t) \end{bmatrix}}_{=\mathrm{y}}$$

很容易将这个线性微分方程在 MATLAB 中编译成一个独立的 M 文件 ode.m，见算法 11.1. 这样一个所谓 ode 函数至少有两个输入参数 t 和 $\mathbf{y}$ 并返回 $\dot{\mathbf{y}}$，记为 yd.

算法 11.1 微分方程 ode.m

```
function yd= ode(t,y);
global a b c r;

yd= [y(2); (a*r^2*exp(2*i*t)+b*r*exp(i*t)+c)*y(1)];
```

解微分方程之前我们必须给全局变量 a, b 和 c 赋值. 因为全局变量 r 被当作一个参量，它将在后面求解时被分别赋值.

```
>> global a b c r
>> a= 1; b= 0.5; c= -4/9;
```

为积分我们利用函数 ode45,MATLAB 的改进的 4/5 阶 Runge-Kutta-Fehlberg 算法，它在前面已被用于 MAPLE 求解.

```
>> options= odeset('RelTol',1e-8,'AbsTol',1e-8);
>>
>> % first set of parameters
>> r= 1;
>> [t1,p1]= ode45('ode',[0 6*pi],[-0.563 0.869*i],options);
>>
>> % second set of parameters
>> r= 0.6;
>> [t2,p2]= ode45('ode',[0 6*pi],[-0.944 0.658*i],options);
```

为获得好的打印质量，相对和绝对误差设为 10^{-8}. 执行函数 ode45 需要四个参数：第一个是提供给导数向量 $\dot{\mathbf{y}}$ 名为 'ode' 的函数; 第二个是积分范围; 第三个是向量 $\mathbf{y_0} = [U(0)+\mathrm{i}V(0), \dot{U}(0)+\mathrm{i}\dot{V}(0)]^{\mathrm{T}}$; 第四个是在目前状态下的误差 options.

由下面的命令产生解的曲线：

```
>> clf;
>> plot(p1(:,1),'b-');
>> hold on;
>> plot(p2(:,1),'r--');
>> hold off;
```

正如所希望的，这些命令产生了一个几乎与由 MAPLE 获得的图 11.1 完全一样的图形.

参考文献

[1] J. HÄUSER AND C. TAYLOR, *Numerical Grid Generation in Computational Fluid Dynamics*, Pineridge Press, Swansea, U.K., 1986.

[2] J. HEINHOLD AND U. KULISCH, *Analogrechnen, BI-Hochschultaschenbücher Reihe Informatik*, Bibliographisches Institut Mannheim /Zurich, 168/168a, 1968.

[3] W. F. HUGHES AND J. A. BRIGHTON, *Fluid Dynamics*, Schaum's Outline Series, McGraw-Hill, USA, 1967.

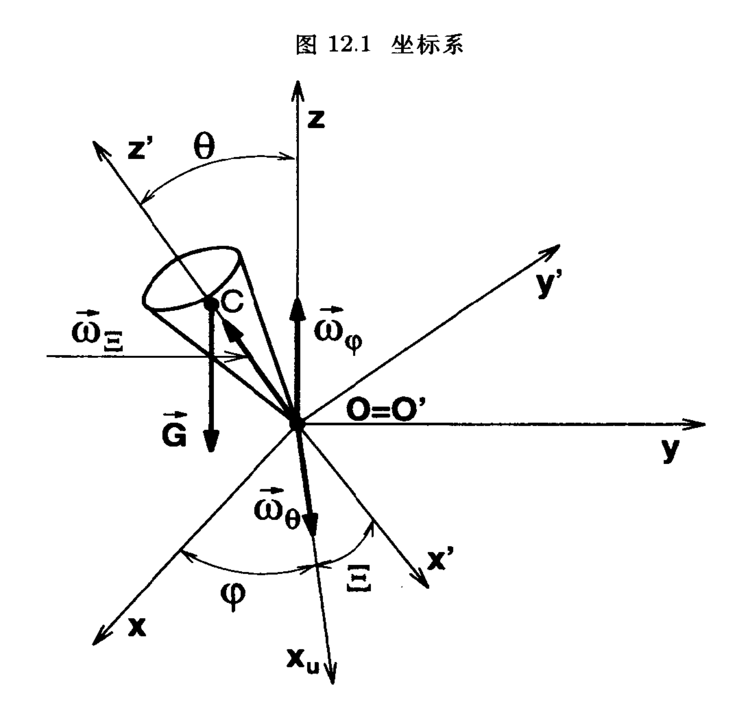
第十二章　陀螺

F. Klvaňa

12.1　引言

本章研究人们熟知的一种儿童玩具 — 陀螺的运动. 从物理的角度, 我们可以把它看成在各向同性引力场中的刚性转子. 设 O 是陀螺尖顶的点.

图 12.1　坐标系



设 (x,y,z) 是原点在转子的 O 点的惯性坐标系, 其中 z 轴的方向为垂直方向 (参见图 12.1). 设 $\vec{G}=(0,0,-mg)$ 是转子的重量 (作用在其质心上), 这里 m 是转子的质量, 而 g 是重力加速度. 则当角速度为 $\vec{\omega}=(\omega_1,\omega_2,\omega_3)$ 时, 这个转子的动能是

$$T=\frac{1}{2}\sum_{i,j=1}^{3} I_{ij}\omega_i\omega_j.$$

注意向量 $\vec{\omega}$ 的方向是其旋转轴的方向. I_{ij} 是转子的惯性张量 (参见 [1,2]). 至少存在一个满足 $O'=O$ 且其惯性张量是对角的固定在物体上的坐标系 (x',y',z'). 在这个坐标系中, 其坐标轴称为惯性主轴. 对于对称转子, 其对称轴是主轴之一, 我们将它标为 z' 轴. 其它两个主轴在与 z' 轴垂直的 (x',y') 平面内. 这时动能具有形式

$$T=\frac{1}{2}I_1\left(\omega_{x'}^2+\omega_{y'}^2\right)+\frac{1}{2}I_3\omega_{z'}^2, \tag{12.1}$$

其中 I_1 和 I_3 是相应的主转动惯量.

使用 Euler 角 (Φ,Θ,Ξ) 描述围绕固定点 O 的旋转是便利的. 我们用 Euler 角来表述经过一系列旋转所得到的由坐标系 (x,y,z) 到 (x',y',z') 的变换 (参见图 12.1):

1. 绕 z 轴旋转角度 $\Phi(x \to x_u)$,
2. 绕 x_u 轴 (称为节线) 旋转角度 $\Theta(z \to z')$,
3. 绕 z' 轴旋转角度 $\Xi(x_u \to x')$.

由坐标系 (x,y,z) 变换到坐标系 (x',y',z') 的变换矩阵 $\hat{A}(\Phi,\Theta,\Xi)$ 被表述为旋转矩阵的乘积

$$\hat{A}(\Phi,\Theta,\Xi) = \hat{R}_z(\Xi)\cdot\hat{R}_x(\Theta)\cdot\hat{R}_z(\Phi), \tag{12.2}$$

其中 $\hat{R}_x$ 和 $\hat{R}_z$ 分别代表关于 x 轴或 z 轴的旋转矩阵.

Euler 角 $(\Phi(t),\Theta(t),\Xi(t))$ 可以用作转子的广义坐标. 设转子在时间间隔 $\mathrm{d}t$ 内关于瞬时轴旋转角度 $|\vec{\omega}|\mathrm{d}t$. 这个旋转可依次表述为分别关于 z 轴, x_u 轴和 z' 轴旋转角度 $|\vec{\omega}_\Phi|\mathrm{d}t = \dot{\Phi}\mathrm{d}t$, $|\vec{\omega}_\Theta|\mathrm{d}t = \dot{\Theta}\mathrm{d}t$ 和 $|\vec{\omega}_\Xi|\mathrm{d}t = \dot{\Xi}\mathrm{d}t$ 的旋转. 我们写成

$$\vec{\omega} = \vec{\omega}_\Phi + \vec{\omega}_\Theta + \vec{\omega}_\Xi. \tag{12.3}$$

角速度 $\vec{\omega}_\Phi,\vec{\omega}_\Theta,\vec{\omega}_\Xi$ 在旋转坐标系 (x',y',z') 中具有下面的分量:

$$\begin{aligned}
((\omega_\Phi)_{x'},(\omega_\Phi)_{y'},(\omega_\Phi)_{z'}) &= \hat{A}(\Phi,\Theta,\Xi)\cdot(0,0,\dot{\Phi})\\
((\omega_\Theta)_{x'},(\omega_\Theta)_{y'},(\omega_\Theta)_{z'}) &= \hat{R}_z(\Xi)\cdot(\dot{\Theta},0,0)\\
((\omega_\Xi)_{x'},(\omega_\Xi)_{y'},(\omega_\Xi)_{z'}) &= (0,0,\dot{\Xi}).
\end{aligned}$$

利用 (12.1) 和 (12.3), 我们把动能 T 表示为广义坐标 $(\Phi(t),\Theta(t),\Xi(t))$ 及其速度 $(\dot{\Phi}(t),\dot{\Theta}(t),\dot{\Xi}(t))$ 的函数

$$T = \frac{1}{2}I_1(\dot{\Theta}^2 + \dot{\Phi}^2\sin^2\Theta) + \frac{1}{2}I_3(\dot{\Xi} + \dot{\Phi}\cos\Theta)^2.$$

关于转子的势能, 我们有

$$V = mgl\cdot\cos\Theta \tag{12.4}$$

其中 l 是转子的质心 C 和定点 O 之间的距离.

我们的问题极适于用 Lagrange 理论求解. 设 $L = T - V$ 是 Lagrange 算子, $(q_1,\cdots,q_n)$ 和 $(\dot{q}_1,\cdots,\dot{q}_n)$ 是具有 n 个自由度系统的广义坐标和速度. 那么, 其运动方程是 Lagrange 方程

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial l}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_i} = 0 \qquad i = 1,\cdots,n.$$

如果 L 不依赖于某个坐标 q_k(这样的坐标称为循环坐标), 则相应的 Lagrange 方程是

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_k}\right) = 0 \Longrightarrow \frac{\partial L}{\partial \dot{q}_k} = p_k = const, \tag{12.5}$$

即, 共轭动量 p_k 是一个运动常数.

由于三个 Lagrange 方程所构成的方程组的相对复杂性, 我们用 MAPLE 对其作推导和研究.

12.2 公式表述和解的基本分析

用 MAPLE 求解问题的过程中，我们使用软件包 linalg 中的向量表示以及下面的变量记号：

$$(\Phi,\Theta,\Xi) \rightarrow Euler := [\Phi,\Theta,\Xi]$$
$$(\dot{\Phi},\dot{\Theta},\dot{\Xi}) \rightarrow dEuler := [\phi,\theta,\xi]$$

```
> with(linalg):
> Euler := vector([Phi, Theta, Xi]):
> dEuler := vector([phi, theta, xi]):
```

作为求解的第一步，我们用 Euler 角来表述 Lagrange 算子．为了表述关于坐标轴 x 和 z 轴的旋转矩阵 R_x 和 R_z，我们定义矩阵算子：

```
> Rz := matrix([[cos, sin, 0], [-sin, cos, 0], [0, 0, 1]]);
> Rx := matrix([[1, 0, 0], [0, cos, sin], [0, -sin, cos]]);
```

$$Rz := \begin{bmatrix} \cos & \sin & 0 \\ -\sin & \cos & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Rx := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos & \sin \\ 0 & -\sin & \cos \end{bmatrix}$$

变换矩阵 A 由方程 (12.2) 给出，

```
> A := evalm(Rz(Xi(t)) &* Rx(Theta(t)) &* Rz(Phi(t))):
```

使用方程 (12.3) 和 (12.4) 可得在坐标系 (x',y',z') 中的角速度向量 $\vec{\omega}$ 为

$$(\omega_{x'},\omega_{y'},\omega_{z'}) = A\cdot(0,0,\dot{\Phi}) + R_z(\Xi)\cdot(\dot{\Theta},0,0) + (0,0,\dot{\Xi}).$$

```
>              # vectors of angular velocities about Euler axis
> omega[Phi] := vector([0,0,phi(t)]):
>              #rotation about axis z - phi
> omega[Theta] := vector([theta(t),0,0]):
>              #rotation about the 1.  xu - theta
>              # vector of the resulting angular velocity
>              # Omega in (x',y',z')
> Omega := evalm(A &* omega[Phi] + Rz(Xi(t)) &* omega[Theta]
>          + vector([0,0,xi(t)]));
```

$$\Omega := [\sin(\Xi(t))\sin(\Theta(t))\,\phi(t) + \cos(\Xi(t))\,\theta(t),\ \cos(\Xi(t))\sin(\Theta(t))\,\phi(t) - \sin(\Xi(t))\,\theta(t),$$
$$\cos(\Theta(t))\,\phi(t) + \xi(t)]$$

那么，由公式 (12.1) 和 (12.4), 可以把 Lagrange 算子写为

$$L = T - V = \frac{1}{2}I_1(\omega_{x'}^2 + \omega_{y'}^2) + \frac{1}{2}I_3\omega_{z'}^2 - mgl\cos\Theta.$$

```
> T := 1/2*I1*(Omega[1]^2 + Omega[2]^2) + 1/2*I3*Omega[3]^2;
>                                #kinetic energy
```

$$T := \frac{1}{2} I1\,((\sin(\Xi(t))\sin(\Theta(t))\,\phi(t) + \cos(\Xi(t))\,\theta(t))^2 + (\cos(\Xi(t))\sin(\Theta(t))\,\phi(t) - \sin(\Xi(t))\,\theta(t))^2) + \frac{1}{2} I3\,(\cos(\Theta(t))\,\phi(t) + \xi(t))^2$$

```
> V := M*g*L*cos(Theta(t)):    #potential energy
> LF := T - V:                 # Lagrangian
```

第二步是研究转子的 Lagrange 方程. 为此，定义 MAPLE 函数

`LEq2(LagrFce, var, dvar, indep)`

是有用的 (参见算法 12.1), 其中 `LagrFce` 是 Lagrange 算子， `var` 和 `dvar` 广义变量和速度的向量，`indep` 是自变量 (时间 t). 在循环坐标的情形，这个函数返回方程 (参见 12.5)

$$\frac{\partial L}{\partial \dot{q}_k} = IMq_k,$$

其中 IMq_k(q_k 的共轭动量) 是运动常数. 这个函数返回一个方程集.

由于标准函数 `diff(g,u)` 仅当 `u` 是变量名时才求导，我们将定义广义的求导函数 `sdiff(g,u)`, 其中 `u` 可以是个表达式.

为了连接字符串 IM 和循环变量名 q_k(由我们定义的一个函数 $q_k(t)$), 我们应该定义连接名称 `name` 和函数的名称 `fce` 的函数 `cname(name,fce)`. 这些函数的定义在算法 (12.1) 中给出.

既然坐标 Φ 和 Ξ 是循环的，函数 `LEq2` 返回运动常数 p_Φ(名称为 IM_Φ) 和 $p_\Xi(IM_\Xi)$ 的两个方程和对应于变量 Θ 的 Lagrange 方程. 为了比较容易选取这些方程的右端，我们把这组方程变换成一个表，其中左端 $(IM_\Phi, IM_\Xi, 0)$ 是相应右端的标记.

```
>              # generation of the Lagrange equation, or integrals
>              # of motion (for Phi and Xi)
> LEqn := LEq2(LF,Euler(t),dEuler(t),t):
> LEqn := table([op(LEqn)]);
>              #transform the result from set to table
```

$LEqn :=$ table([

$$IM_\Phi = I1\,\phi(t) - I1\,\phi(t)\cos(\Theta(t))^2 + I3\cos(\Theta(t))^2\,\phi(t) + I3\cos(\Theta(t))\,\xi(t)$$

$$IM_\Xi = I3\cos(\Theta(t))\,\phi(t) + I3\,\xi(t)$$

$$0 = I1\,(\frac{\partial}{\partial t}\theta(t)) - I1\cos(\Theta(t))\,\phi(t)^2\sin(\Theta(t)) + I3\sin(\Theta(t))\,\phi(t)^2\cos(\Theta(t)) + I3\sin(\Theta(t))\,\phi(t)\,\xi(t) - M\,g\,L\sin(\Theta(t))$$

])

最重要的运动常数是能量 $E = T + V$. 因为 $\omega_{z'}$ 与运动常数 p_{Ξ} 成比例, 所以量 $E_c = E - 1/2 I_3 \omega_{z'}^2$, 也是一个运动常数. 在以后的讨论中, 我们将使用 E_c 而不是 E.

```
>             # Integrals of motion:  IM[Phi], IM[Xi] and energy Ec
>  Ec := simplify(T + V- 1/2*I3*Omega[3]^2);
```

$$Ec := \frac{1}{2}\, I1\, \phi(t)^2 - \frac{1}{2}\, I1\, \phi(t)^2 \cos(\Theta(t))^2 + \frac{1}{2}\, I1\, \theta(t)^2 + M\, g\, L \cos(\Theta(t))$$

算法 12.1 生成 Lagrange 方程的函数

函数 sdiff

```
sdiff := proc(u,svar) local p;
     #differentiation of u with respect to expression svar
     subs(p = svar,diff(subs(svar = p,u),p))
end;
```

函数 cname

```
cname:= proc (nam,fce) local p;
    #add name of function  fce  to the name  nam
    if type(fce,function) then
        p:=op(0,fce); nam[p]
     else
        nam.fce
     fi
end;
```

函数 LEq2

```
LEq2 := proc (LagrFce, var::vector,dvar::vector,indep::name)
          local  i,j,N,res;
   #the generation of the Lagrange equations or constants
   # of motion IM (for cyclic variables) for the Lagrange
   # function LagrFce and vector of generalized coordinates
   # var and velocities dvar and independent variable indep
   N:=vectdim(var);
   for i to N do
       res[i]:=simplify(sdiff(LagrFce,var[i]));
       if res[i]=0 then
           res[i] := cname('IM',var[i])
                     = sdiff(LagrFce, dvar[i])
       else
           res[i] := 0 = diff(sdiff(LagrFce, dvar[i]), indep)
                          - res[i]
       fi;
    od;
    {seq(simplify(res[i]), i=1..N)}
```

```
    end;
```

为了简捷，我们将执行下面的步骤：

1. 由 p_Φ 和 p_Ξ 的方程解出未知量 $[\dot\Phi,\dot\Xi]$, 然后将之代入 E_c.
2. 引入参数
$$b = \frac{p_\Phi}{I_1}, a = \frac{p_\Xi}{I_1}, \beta = \frac{1}{2}\frac{I_1}{mgl}, \alpha = \frac{2E_c}{I_1}$$
   并由 E_c 的方程解出未知量 $\dot\Theta^2$.
3. 将 $u = \cos\Theta$ 代入所得的方程中.

结果我们得到下面关于 $u(t)$ 的微分方程，它描述了转子的所谓章动 (nutation).

$$\dot u^2 = y(u) := \beta u^3 - (\alpha + a^2)u^2 + (2ab - \beta)u + \alpha - b^2. \tag{12.6}$$

由 $y(u) > 0$ 得到：

1. 由 $\beta > 0$ 可得 $y(\pm\infty) = \pm\infty$ 且 $y(\pm 1) \le 0$; 因此 $y(u)$ 至少有一个根 $u > 1$.
2. 由 $y(\pm 1) < 0$, $y(u)$ 在区间 $[-1, 1]$ 内有两个根或没有根. 对于物理的实际条件，应该存在两个根： $u_1 \le u_2 \in [-1, 1]$. 因为对于 $\cos\Theta_1 = u_1$ 和 $\cos\Theta_2 = u_2$, $\dot\Theta = 0$, 所以 $\Theta(t) \in [\Theta_2, \Theta_1]$, 因而陀螺的章动角在 Θ_1 和 Θ_2 之间摆动.

```
>              # find phi, xi with subs.  IMXi=a*I1, IMPhi=b*I1
> phixi := solve({LEqn[IM[Phi]] = b*I1, LEqn[IM[Xi]] =a *I1},
>              {phi(t),xi(t)});
```

$$phixi := \{\xi(t) = -\frac{I3\cos(\Theta(t))^2 a - I3\cos(\Theta(t))\, b + a\, I1 - a\, I1\cos(\Theta(t))^2}{(-1 + \cos(\Theta(t))^2)\, I3},$$
$$\phi(t) = \frac{\cos(\Theta(t))\, a - b}{-1 + \cos(\Theta(t))^2}\}$$

```
> Econst := subs(phixi,Ec):
>              # substitution for xi and phi in Ec
> theta2 := solve(subs(M = I1/(2*g*L)*beta,
>              Econst = I1*alpha/2),theta(t)^2):
>              # substitution
>              # cos(Theta) = u -> -sin(Theta)*theta = du
>              # then du2 = diff(u(t),t)^2
> du2 := simplify(subs(cos(Theta(t)) = u,
>        theta2*(1-cos(Theta(t))^2)));
```

$$du2 := \beta u^3 - u^2 a^2 - \alpha u^2 + 2\, u\, a\, b - \beta u - b^2 + \alpha$$

```
>              # analysis of the solution due to the form of du2
> collect(du2,u);
```

$$\beta u^3 + (-a^2 - \alpha)\, u^2 + (2\, a\, b - \beta)\, u - b^2 + \alpha$$

```
> seq(factor(subs(u = i, du2)), i = {-1,1});
```

$$-(a + b)^2,\ -(a - b)^2$$

```
>              # so du2 (+-1) < 0 for b <> 0
```

陀螺关于垂直轴 z 轴的旋转 (所谓进动 procession) 由 $\Phi(t)$ 描述，它是微分方程

$$\dot{\Phi} = \frac{b - a\cos\Theta}{\sin^2\Theta} \tag{12.7}$$

的解 (参见上面变量 phixi[1] 的值). 然后，我们能够根据 $\dot{\Phi}$ 在点 Θ_1, Θ_2(在该处 $\dot{\Theta} = 0$) 的符号对转子的运动类型分类. 设 $\Theta_1 \geq \Theta_2 > 0$, 因而 $u_2 = \cos\Theta_2 > u_1$:

1. 如果 $b/a > u_2$ 或 $b/a < u_1$, 则 $\dot{\Phi}$ 不改变其符号 (参见图 12.2);
2. 如果 $b - au_2$ 和 $b - au_1$ 异号，则 $\dot{\Phi}$ 周期性地改变符号 (参见图 12.3).

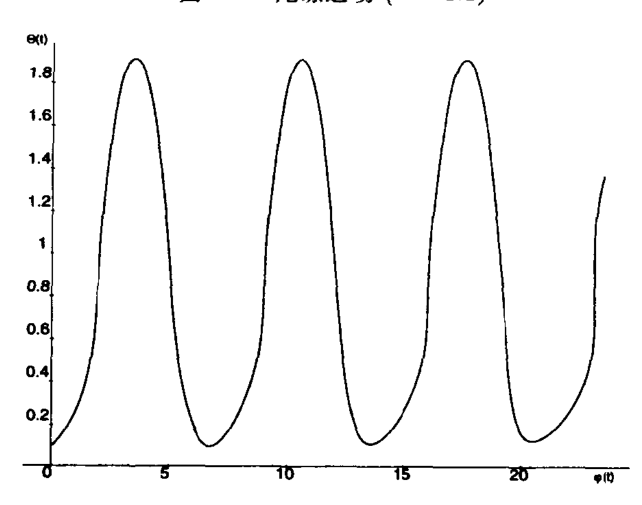
微分方程 (12.6) 可以用椭圆函数解出. 然而，我们将数值求解这个问题. 不过微分方程 (12.6) 的上述形式不适宜用来求数值解.

12.3 数值解

我们将从由 MAPLE 生成的并存储于变量 LEq2 下的 Lagrange 函数开始. 转子运动的最重要特征是由时间的因变量 $\Phi(t)$ 和 $\Theta(t)$(转子的进动和章动角) 给出的，所以我们将只求这些变量.

因为我们求得了对 $\dot{\Phi}$ 和 $\dot{\Xi}$ 的解 (它们仅依赖于 Θ 并存储在 MAPLE 的变量 phixi 中), 所以我们能够在后几个与变量 Θ 相关的 Lagrange 方程中用它们替代 $\dot{\Phi}$ 和 $\dot{\Xi}$. 这个方程与平凡方程 diff(Theta(t),t) = theta(t) 以及来自 phixi 中的 $\dot{\Phi}$(参见 (12.7)) 的方程，构成了关于变量 Phi(t), Theta(t) 和 theta(t) 的含三个一阶微分方程的方程组. 现在我们数值地求解这个方程组.

图 12.2 陀螺运动 ($\lambda = 1.1$)




```
>              # we substitute for xi and phi
>              # in LEqn[0] (DE for Theta)
>  eq[Theta] := simplify(subs(phixi, LEqn[0])):
>  eq[Theta] := simplify(eq[Theta]*I1*(cos(Theta(t))^2 - 1)^2):
>              # simplify constant parameters in DE
```

```
> eq[Theta] := simplify(subs(M = I1/(2*g*L)*beta, eq[Theta]/I1^2));
```

$$eq_\Theta := (\frac{\partial}{\partial t}\theta(t)) - 2(\frac{\partial}{\partial t}\theta(t))\cos(\Theta(t))^2 + (\frac{\partial}{\partial t}\theta(t))\cos(\Theta(t))^4 + \cos(\Theta(t))^2\sin(\Theta(t))\,a\,b$$
$$- \cos(\Theta(t))\sin(\Theta(t))\,b^2 - \sin(\Theta(t))\cos(\Theta(t))\,a^2 + \sin(\Theta(t))\,b\,a - \frac{1}{2}\beta\sin(\Theta(t))$$
$$+ \beta\sin(\Theta(t))\cos(\Theta(t))^2 - \frac{1}{2}\beta\sin(\Theta(t))\cos(\Theta(t))^4$$

```
>            # DE for Phi
> eq[Phi] := diff(Phi(t),t) = subs(phixi, phi(t));
```

$$eq_\Phi := \frac{\partial}{\partial t}\Phi(t) = \frac{\cos(\Theta(t))\,a - b}{-1 + \cos(\Theta(t))^2}$$

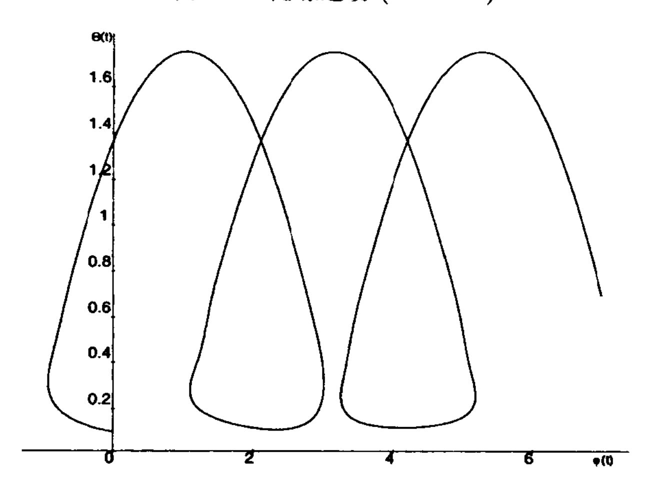
求解的方程组依赖三个参数 a, b 和 β 以及三个初值 $\Phi(0), \Theta(0)$ 和 $\dot{\Theta}(0)$. 注意 $\beta > 0$ 而 a 和 b 能够被取为正值. 我们取初值条件为

$$\Phi(0) = 0, \Theta(0) = \Theta_0, \dot{\Theta}(0) = 0, \qquad \text{其中} \qquad \Theta_0 = 0.1,$$

参数为

$$a = 1, b = \lambda a\cos(\Theta_0),\ \beta = 1, \qquad \text{其中} \qquad \lambda = 1.1.$$

图 12.3 陀螺运动 ($\lambda = 0.96$)



由对方程 (12.7) 的讨论得到两种类型的进动. 如果 $\lambda > 1$, 上面得到的解是 $\dot{\Phi}$ 符号不变, 是纯进动的解. 对于 $\lambda < 1$, 得到的解是 $\dot{\Phi}$ 会变号, 它对应于振荡进动.

由 $\Phi(t)$ 对 $\Theta(t)$ 的参数作图展示这个解. 结果分别在图 12.2($\lambda = 1.1$) 和图 12.3($\lambda = 0.96$) 中给出.

```
>           #d definition of initial conditions
> Theta0 := 0.1:  #Theta(0)
> initc := Theta(0) = Theta0, theta(0) = 0, Phi(0) = 0:
>           # definition of parameters
```

```
> a := 1:  beta := 1:
> lambda := 1.1:  # 0.96 second plot
> b := a*cos(Theta0)*lambda:
>             # dependent variables of the system DE
> var := {Phi(t),Theta(t),theta(t)}:
>             # solving DE
> res := dsolve({eq[Theta] = 0, diff(Theta(t), t) = theta(t),
>        eq[Phi], initc}, var, numeric);
```

$$res := \mathbf{proc}(rkf45_x)\ \ldots\ \mathbf{end}$$

```
>             # resulting plot
> plot([seq([subs(res(i*0.05), Phi(t)),
>             subs(res(i*0.05), Theta(t))], i = 0..300)]);
```

参考文献

[1] HERBERT GOLDSTEIN, *Classical Mechanics*, Addison-Wesley, 1980.

[2] M. R. SPIEGEL, *Theoretical Mechanics*, McGraw-Hill, 1980.

第十三章　标度问题

J.Buchar and J.Hřebíček

13.1　引言

当使用一个压电材料的压力转换器测量一个容器内的气压变化时 (如发动机气缸或枪管), 要得到相当高的精确值, 才能达到特定的绝对精确度. 为此需要特殊的测量和标度技术, 能够定量地确定转换器的动态测量性质. 转换器输出的是电压. 因此我们必须给转换器定标, 才能最后得到压力. 当我们测量的是稳态压力时, 这一点不难做到. 设计能测量动态压力的设备就非常复杂. 这个问题已被人利用水压舱解决, 参见 [3]. 对这个问题我们最近设计了一个实验方法用来解答动态压力标度问题. 这个方法的基本问题是建立一个物理模型, 它可用数学描述水压脉冲. 这个模型使得我们能够用绝对压力单位给一个动态脉冲压力转换器定标. 实验方案如图 13.1 描述. 在下节我们将用 [1] 中的结果设计一个物理模型.

13.2　物理模型的描述

压力转换器利用水压在一个适当的水压舱内确定标度. 某些应用问题, 如测量汽车发动机或枪管内的压力, 必须在动态下确定标度. 为此我们设计了一个如图 13.1 描述的简单实验, 在一个圆柱形的水压舱内注入一种可压缩液体. 液体的初始体积是 V_0, 活塞与圆柱有相同的截面积 S, 且被质量为 M 速度为 $v_0 > 0$ 的物体撞击. 活塞的运动 $x(t)$ 导致体积的变化 $\Delta V = Sx(t)$. 体积变化对应一个压力 $p(t)$, (对几乎所有可压缩液体) 这个压力可以由方程

$$p(t) = \alpha\frac{\Delta V}{V_0} + \beta(\frac{\Delta V}{V_0})^2 \tag{13.1}$$

求得, 其中 $\alpha > 0$ 和 $\beta > 0$ 是所给液体的参数.

设撞击速度 v_0 被限制为某个合理值. 我们可以忽略波动过程. 活塞的移动可以用微分方程描述

$$M\ddot{x} = -p(t)S, \tag{13.2}$$

初始条件为

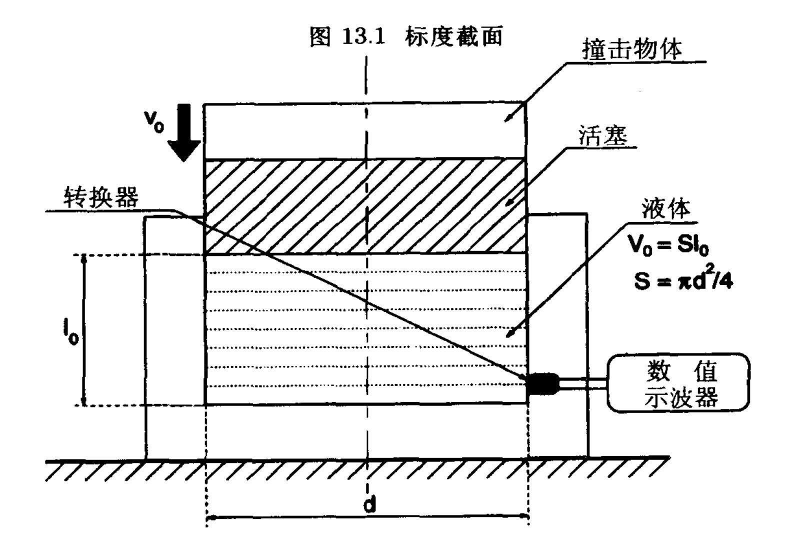
$$\begin{aligned} x(0) &= 0, \\ v = \dot{x}(0) &= v_0. \end{aligned} \tag{13.3}$$

将关于压力 $p(t)$ 的方程 (13.1) 和 $\Delta V = Sx(t)$ 代入方程 (13.2) 得:

```
> M*diff(x(t), t, t) = - (alpha*dV/V0 + beta*(dV/V0)^2)*S;
```

$$M\,(\frac{\partial^2}{\partial t^2}\,\mathrm{x}(t)) = -(\frac{\alpha\,dV}{V0} + \frac{\beta\,dV^2}{V0^2})\,S$$

图 13.1 标度截面




```
> subs(dV = S*x(t), ")/S;
```

$$\frac{M\,(\frac{\partial^2}{\partial t^2}\,\mathrm{x}(t))}{S} = -\frac{\alpha\,S\,\mathrm{x}(t)}{V0} - \frac{\beta\,S^2\,\mathrm{x}(t)^2}{V0^2}$$

用公式 $S = \pi d^2/4$, $V_0 = \pi d^2 l_0/4$ 和常数

$$a = \sqrt{\frac{\alpha\pi}{4}\frac{d^2}{Ml_0}} \qquad \text{和} \qquad b = \frac{\beta\pi}{4}\frac{d^2}{Ml_0^2}$$

简化方程.

```
> subs(S = Pi*d^2/4, V0 = Pi*d^2*l0/4, "):
> simplify(", {a^2*4*M*l0 = alpha*Pi*d^2,
>             b*4*M*l0^2 = beta*Pi*d^2}, [alpha, beta]);
```

$$4\,\frac{M\,(\frac{\partial^2}{\partial t^2}\,\mathrm{x}(t))}{\pi\,d^2} = -4\,\frac{\mathrm{x}(t)\,M\,(a^2 + b\,\mathrm{x}(t))}{\pi\,d^2}$$

```
> eq := "*Pi*d^2/4/M;
```

$$eq := \frac{\partial^2}{\partial t^2}\,\mathrm{x}(t) = -\mathrm{x}(t)\,(a^2 + b\,\mathrm{x}(t))$$

```
> alpha*dV/V0 + beta*(dV/V0)^2:
> subs(dV = S*x(t), S = Pi*d^2/4, V0 = Pi*d^2*l0/4, ");
```

$$\frac{\alpha\,\mathrm{x}(t)}{l0} + \frac{\beta\,\mathrm{x}(t)^2}{l0^2}$$

现在我们可以将初值问题 (13.2)-(13.3) 重新写作

$$\ddot{x}(t) + a^2 x(t) + b x(t)^2 = 0, \tag{13.4}$$

$$\begin{aligned} x(0) &= 0, \\ \dot{x}(0) &= v_0. \end{aligned} \tag{13.5}$$

利用上面初值问题的解 $x(t)$, 我们可以计算体积的变化 $\Delta V = Sx(t)$. 从 (13.1) 我们得到 t 时刻的压力值 $p = p(t)$

$$p(t) = \frac{x(t)}{l_0}(\alpha + \beta\frac{x(t)}{l_0}). \tag{13.6}$$

在测量水压脉冲时我们用数值示波器记录了转换器的电压 $U = U(t)$. 对某个时间区间 $T = [0, t_{max})$ 我们获得转换器的实验记录, 参见图 13.1. 对同样这个时间区间我们必须求以上初值问题的解 $x(t)$. 然后我们可以比较实验得到的水压与从 (13.6) 计算得到的压力 $p(t)$. 为特殊的目的, 只需在压力达到其最大值的子区间 $T_m \subset T$ 上比较测量的和计算的压力.

当 $p(t) = kU(t)$, k 是常数时, 转换器的定标非常简单. 如果 k 不是常数, 我们必须用 Fourier 变换方法. 这种方法的详细分析超出了本章的范围.

13.3 解的分解逼近

假设初值问题 (13.4)-(13.5) 的解 $x(t)$ 具有形式

$$x(t) = x_a(t) + x_b(t),$$

其中 $x_a(t)$ 是自由无阻尼运动 [4] 的初值问题的解

$$\begin{aligned} \ddot{x}_a(t) + a^2 x_a(t) &= 0, \\ x_a(0) &= 0, \\ \dot{x}_a(0) &= v_0. \end{aligned}$$

当 $a \neq 0$ 上面的初值问题有解析解

$$x_a(t) = \frac{v_0 \sin(at)}{a}. \tag{13.7}$$

注意到可以用 MAPLE 推导这个解:

```
> deq := {diff(xa(t), t, t) + a^2*xa(t) = 0, xa(0) = 0, D(xa)(0) = v0}:
> dsolve(deq, xa(t));
```

$$\mathrm{xa}(t) = \frac{v0 \sin(a\,t)}{a}$$

初值问题 (13.4)-(13.5) 的解 $x(t)$ 具有形式

$$x(t) = \frac{v_0 \sin(at)}{a} + x_b(t), \tag{13.8}$$

假设 $x_b(t)$ 是 $x_a(t)$ 的小扰动. 将 $x(t)$ 代入 (13.4) 我们得到关于 $x_b(t)$ 的初值问题

$$\begin{aligned} \ddot{x}_b(t) + a^2 x_b(t) + b x_b(t)^2 + \tfrac{2bv_0 \sin(at)}{a} x_b(t) + b(\tfrac{v_0 \sin(at)}{a})^2 &= 0, \\ x_b(0) &= 0, \\ \dot{x}_b(0) &= 0. \end{aligned}$$

我们将用 MAPLE 求这个微分方程的解 $x_b(t)$, 用它解微分方程的有力工具 — 截断幂级数. 这里截断幂级数在 $t = 0$ 展开. 它被用于阶数小于或等于截断幂级数的阶数 N 的多项式. MAPLE 的指令如下 (我们将令阶数 =12)

```
>              # initial value problem B
> eqb := diff(xb(t), t, t) + a^2*xb(t) + b*xb(t)^2
>         + 2*b*v0*sin(a*t)*xb(t)/a + b*(v0*sin(a*t)/a)^2 = 0;
```

$$eqb := (\frac{\partial^2}{\partial t^2}\,\mathrm{xb}(t)) + a^2\,\mathrm{xb}(t) + b\,\mathrm{xb}(t)^2 + 2\,\frac{b\,v0\,\sin(a\,t)\,\mathrm{xb}(t)}{a} + \frac{b\,v0^2\,\sin(a\,t)^2}{a^2} = 0$$

```
> incondb := xb(0) = 0,D(xb)(0) = 0:
>         # determination of Order truncated power series
> Order := 12:
> solb := dsolve({eqb, incondb}, xb(t), series);
```

$$solb := \mathrm{xb}(t) = -\frac{1}{12}\,b\,v0^2\,t^4 + \frac{1}{72}\,b\,v0^2\,a^2\,t^6 + \frac{1}{252}\,b^2\,v0^3\,t^7 - \frac{1}{960}\,b\,v0^2\,a^4\,t^8 - \frac{5}{6048}\,b^2\,v0^3\,a^2\,t^9 + \frac{1}{362880}\,b\,v0^2\,(17\,a^6 - 60\,b^2\,v0^2)\,t^{10} + \frac{1}{12320}\,a^4\,b^2\,v0^3\,t^{11} + \mathrm{O}(t^{12})$$

```
> polb := convert(rhs(solb), polynom):
```

由 Maple 给出的截断幂级数 $s_N(t)$ 的多项式逼近等价于

$$p_N(t) = -\frac{bv_0^2t^4}{12}\Big(1 - \frac{a^2t^2}{6} - \frac{v_0bt^3}{21} + \frac{a^4t^4}{80} + \frac{5v_0a^2bt^5}{504} + \frac{v_0^2t^6b^2}{504} - \frac{17t^6a^6}{30240} - \frac{3ba^4v_0t^7}{3080}\cdots\Big). \tag{13.9}$$

用逼近解 $\tilde{x}_N(t) = x_a(t) + p_N(t)$, 将 $\tilde{x}_N(t)$ 代入方程 (13.6) 可得压力 $p(t)$ 的逼近 $\tilde{p}_N(t)$.

为获得 $x(t)$ 的充分精确的逼近要事先选择多项式的阶数，我们必须估计截断幂级数的收敛域. 为此我们将进行一些简单的富有启发性的讨论. 从解 $x_a(t)$ 和压力 $p(t)$ 的定义可知压力脉冲在

$$t < t_a = \frac{\pi}{a} \tag{13.10}$$

时是正的. 这是确定区间 T 的第一个条件，在这个区间上我们求解 $x(t)$ 和压力 $p(t)$.

设 (13.9) 式括号中的每一项绝对值均小于 1. 我们可以看到 (13.9) 式括号中的第四项和它以后的项是第二项和第三项乘以某个绝对值小于 1 的常数的组合，所以只有第二项和第三项起作用. 这个假设导出了第二个条件

$$t < t_{coef} = \min(|\frac{a}{\sqrt{6}}|^{-1}, |\frac{v_0b}{21}|^{-1/3}). \tag{13.11}$$

显然对这些 t 的值，在 (13.9) 式括号中每项的绝对值均小于 1. 如果条件 (13.11) 有效，那么条件 (13.10) 成立. 由 (13.7) 得 $|x_a(t)| \le v_0/a$, 且当压力 $p(t) > 0$ 标度才有意义. 为了获得解 $x(t)$ 的正逼近 $\tilde{x}_N(t)$, 从 (13.7)(13.8) 和 (13.11), 我们引入第三个条件

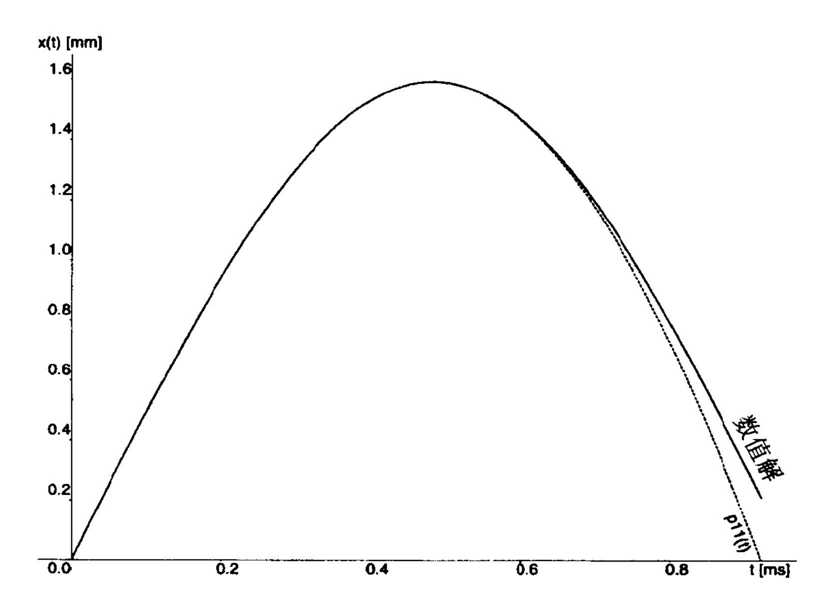
$$t < t_b = (\frac{abv_0}{12})^{-1/4}. \tag{13.12}$$

条件 (13.11) 和 (13.12) 确定了最大区间 $[0, t_m)$, 其中 $t_m = \min(t_{coef}, t_b)$. 于是截断幂级数将收敛到解 $x_b(t)$, 且逼近解 $\tilde{x}_N(t)$ 将收敛到我们问题的解 $x(t)$.

为达到给定的精确度，我们该如何选择截断幂级数的阶数？我们将采用下面的标准

$$|a_N| + |a_{N-1}| \le \varepsilon|\sum_{i=0}^{N} a_i|, \tag{13.13}$$

图 13.2 解的数值和多项式逼近



其中 a_i 是幂级数的项，ε 为可接受的局部相对逼近误差的最大值 [2]. 这个标准利用了 N 阶幂级数展式的最后两项绝对值的和作为截断误差的量度，将它与截断幂级数绝对值联系起来. 在此，当幂级数阶数为 12 时，从标准 (13.13) 得 $\varepsilon = O(10^{-3})$

例子

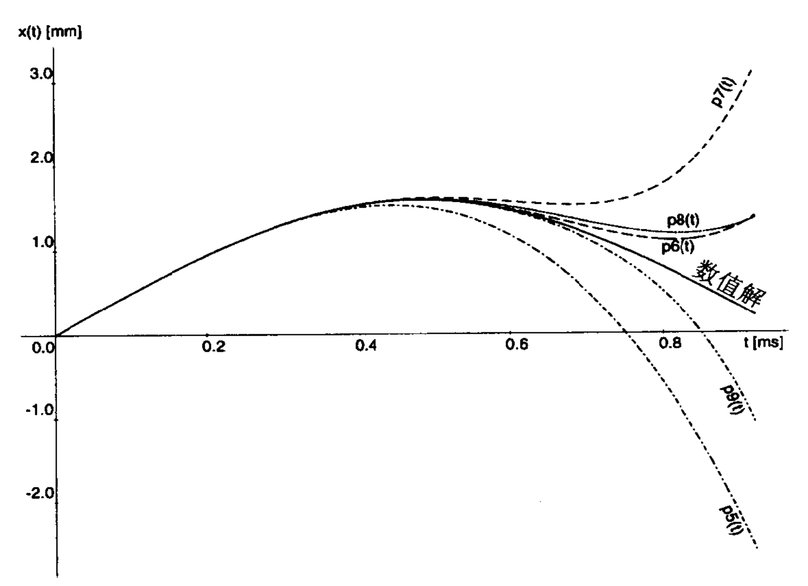
为证实这种方法，我们将解上面的初值问题，并采用 J.Buchar 得到的实验数据 $v_0 = 5[ms^{-1}]$, $\alpha = 3.8E9[Nm^{-2}]$, $\beta = 3E10[Nm^{-2}]$, $l_0 = 2E-2[m]$, $d = 1.5E-2[m]$, $M = 5[kg]$. 下面的 MAPLE 程序将绘出上面初值问题的数值解和用不同阶数截断幂级数的多项式逼近解的图形.

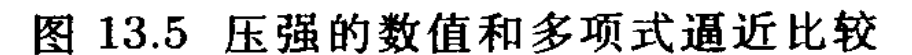
```
>           # experimental constants
> alpha := 3.8*10^9:  beta := 30*10^9:  d := 0.015:
> l0 := 0.02:  M := 5:  v0 := 5:
>           # determination of parameter a, b
>  a := sqrt(alpha*Pi*d^2/(4*M*l0)):  b := beta*Pi*d^2/(4*M*l0^2):
>           # initial value problem B
>  eqb := diff(xb(t), t, t) + a^2*xb(t) + b*xb(t)^2
>             + 2*b*v0*sin(a*t)*xb(t)/a + b*(v0*sin(a*t)/a)^2=0:
> incondb := xb(0) = 0, D(xb)(0) = 0:
>           # numerical solution
> numsol := dsolve({eqb, incondb}, xb(t), numeric):
> sol := x -> subs(numsol(x), xb(t) + v0*sin(a*t)/a):
> solp := t-> (sol(t)/l0)*(alpha + beta*(sol(t)/l0))/10^9:
>           # estimation of radius of convergence
> tm := min(evalf(sqrt(6)/a),evalf(21/(v0*b))^(1/3),
>               evalf(21/(a*b*v0))^(1/4)):
```

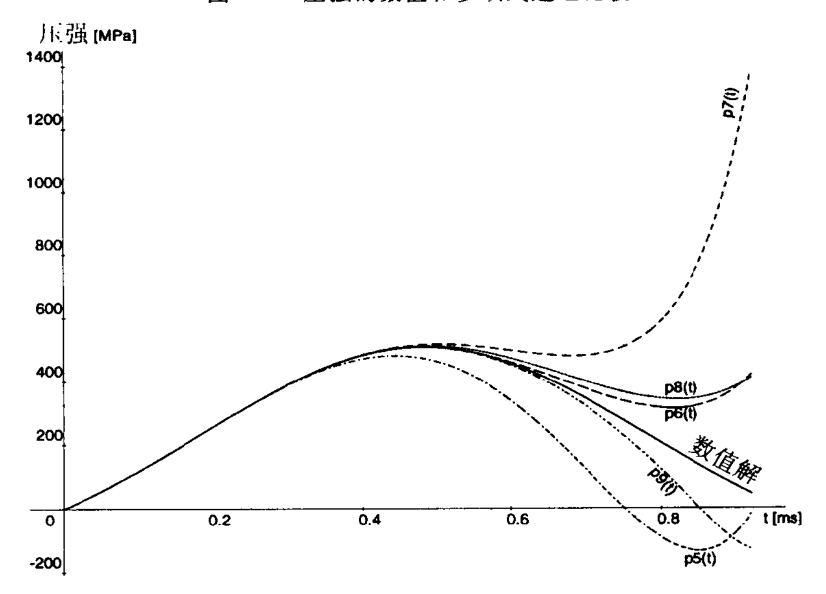
图 13.3 压强的数值和多项式逼近



图 13.4 解的数值和多项式逼近比较




图 13.5 压强的数值和多项式逼近比较




```
>           # saving graphs of numerical solution
> X[num] := plot(sol, 0..tm, thickness=3):
> P[num] := plot(solp, 0..tm, thickness=3):
>           # analytic solution of first equation
> xa := v0*sin(a*t)/a:
>           # Order of truncated power series approximation
>           # and preparation of graphs
> ORD1 := [seq(i, i = 6..10), 12]:
> ORD2 := [seq(i, i = 6..10)]:
>           # polynomial approximation of 5th - 9th
>           # and 11th degree of solution
> for ord in ORD1 do:
>     Order := ord:
>     sersol := dsolve({eqb, incondb}, xb(t), series):
>     pol := convert(rhs(sersol), polynom):
>     sol := pol + xa:
>     solp := (sol/10)*(alpha + beta*sol/10)/10^9:
>           # saving graphs
>     X[ord] := plot(sol, t = 0..tm):
>     P[ord] := plot(solp, t = 0..tm):
> od:
>           # display graphs of solutions and pressures
>           # numeric and acceptable power series solution
> with(plots):
> display({X[num], X[12]});
> display({P[num], P[12]});
>           # comparison of numeric and polynomial solution
```

```
> display({X[num], seq(X[i], i = ORD2)});
> display({P[num], seq(P[i], i = ORD2)});
```

13.4 结论

从这些图形我们可以看到阶数小于 8 的多项式是不可接受的，因为在区间的末端精确度很差. 但在一个较小区间内标度 (从零到压力的最大值) 阶数高于 5 的多项式是可接受的. 为得到对问题解 $x(t)$ 的精确逼近，根据所给实验数据，阶数为 11 的多项式逼近效果最好.

参考文献

[1] M. W. Chang and W. M. Zhu, *Dynamic calibration of chamber pressure measuring equipment*, Proceedings 13th international symposium on ballistics, 1, 1992, pp. 443-450.

[2] H. J. Halin, *The applicability of Taylor series methods in simulation*, Proceedings of 1983 Summer Computer Simulation Conference, 2, 1983,pp. 1032-1076.

[3] G. Resch, *Dynamic conformity and dynamic peak pressure accuracy – to new features of pressure transducers*, Proceeding 14th Transducer Workshop, 1987, pp. 1-10.

[4] D. G. Zill and M. R. Cullen, *Advanced Engineering Mathematics*, PWSKENT, Boston, 1992.

第十四章　热流问题

S. Bartoň and J. Hřebíček

14.1　引言

热流问题是热力学的一个重要组成部分．这些问题的解影响许多其它技术问题．描述热流规律的最重要的方程是热方程 (Fourier 方程)

$$a^2\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right)=\frac{\partial T}{\partial t}. \tag{14.1}$$

解方程 (14.1) 的难度依赖于边界和初始条件．我们将热流问题分成两类．

1. 定态问题，此时温度 T 不是时间 t 的函数，方程 (14.1) 变成 Laplace 方程．在这一章的第一部分我们将解这个问题．
2. 时间依赖问题，这些问题通常用数值方法求解．我们将证明，对非常简单的边界条件 $T(0,t)=$ 常数，和初始条件 $T(x,0)=$ 常数的一维问题 $T=T(x,t)$ 可以找到一个解析解．

14.2　球壳上的热流

考虑一个空心的球，内半径为 r_1, 外半径为 r_2. 设 T_1 和 T_2 分别为内外球面上的不变温度， k 是球壳的热导系数．我们将求出热流的定态解和温度在径向 r 上的分布．

设 x,y,z 是欧氏空间的坐标．如果球体的中心在原点 $O(0,0,0)$, 则温度 $T=T(x,y,z)$ 是球体内位置的函数．由对称性和边界条件，显然对球面 $x^2+y^2+z^2=r^2, r_1\le r\le r_2$ 上所有点 (x,y,z), $T(x,y,z)=$ 常数．因此 $T=T(r)$ 仅是 r 的函数，我们的问题是一维的．在 MAPLE 中我们将用两种方法解答它，两种方法的区别在于物理模型不同．

14.2.1　定态的热流模型

设 $\dot{q}$ 是热流密度．根据 Fourier 定律

$$\dot{q}=-k\ gradT \qquad [Wm^{-2};Wm^{-1}K^{-1},K]. \tag{14.2}$$

考虑中心在原点，半径为 $r, r_1\le r\le r_2$ 的球面，其面积为 $A=4\pi r^2$. 则穿过它的热流 $\dot{Q}$ 为

$$\dot{Q}=\dot{q}A \qquad [W]. \tag{14.3}$$

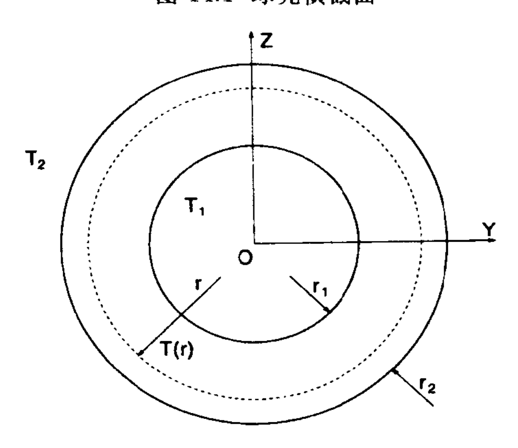
这个流量的定态不依赖半径 r, 因为在球体内没有热源．对一维问题， $grad\,T$ 可以由 $\mathrm{d}T/\mathrm{d}r$ 代替．于是从 (14.2) 和 (14.3) 我们得到

$$\dot{Q}=-k4\pi r^2\frac{\mathrm{d}T}{\mathrm{d}r}. \tag{14.4}$$

记

$$a=\frac{\dot{Q}}{4k\pi} \tag{14.5}$$

图 14.1 球壳横截面



则有

$$\frac{\mathrm{d}T}{\mathrm{d}r} = -a\frac{1}{r^2}, \tag{14.6}$$

其中参数 a 是未知的. 这个方程的一般解是参数 a 的函数, 并且可手工推导得到.

$$T = T(r; a) \tag{14.7}$$

现在我们说明解析地求解方程 (14.6) 是多么简单:

```
> sol := dsolve(diff(T(r), r) = -a/(r^2), T(r));
```

$$sol := \mathrm{T}(r) = \frac{a + _C1\, r}{r}$$

```
> T := unapply(rhs(sol), r);
```

$$T := r \to \frac{a + _C1\, r}{r}$$

我们应该用初始条件确定 a 和 $_C1$

$$T(r_1) = T_1, \qquad T(r_2) = T_2, \tag{14.8}$$

```
> sol := solve({T(r1) = T1, T(r2) = T2}, {a, _C1});
```

$$sol := \{_C1 = \frac{-T2\, r2 + T1\, r1}{-r2 + r1}, a = \frac{r2\, r1\, (-T1 + T2)}{-r2 + r1}\}$$

```
> assign(sol);
> normal(T(r));
```

$$\frac{-r1\, r2\, T1 + r1\, r2\, T2 - r\, T2\, r2 + r\, T1\, r1}{(-r2 + r1)\, r}$$

现在温度函数完全得到了. 对给定的外侧和内侧表面的温度, 球壳内的温度仅是 r 的函数.

最后, 我们可以用 (14.5) 根据参数 a 的定义计算所求热流 $\dot{Q}$.

```
> Q := 4*k*Pi*a;
```

$$Q := 4\,\frac{k\,\pi\, r2\, r1\, (-T1 + T2)}{-r2 + r1}$$

14.2.2 定态的 Fourier 模型

对二维定态问题我们有

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=0, \tag{14.9}$$

或用极坐标 r,φ 形式

$$\frac{\partial^2 T}{\partial r^2}+\frac{2}{r}\frac{\partial T}{\partial r}+\frac{cos\varphi}{r^2 sin\varphi}\frac{\partial T}{\partial \varphi}+\frac{1}{r^2}\frac{\partial^2 T}{\partial \varphi^2}=0. \tag{14.10}$$

因为，此时 $T=T(r)$ 仅是 r 的函数，方程 (14.10) 简化为

$$\frac{\partial^2 T}{\partial r^2}+\frac{2}{r}\frac{\partial T}{\partial r}=0 \tag{14.11}$$

具有初值条件 (14.8). 解 (14.11) 可得温度 $T(r)$, 它为半径 r 的函数:

```
> restart;
> inicon := T(r1) = T1, T(r2) = T2:
> deq := diff(T(r), r, r) + diff(T(r) ,r)*2/r = 0;
```

$$deq := (\frac{\partial^2}{\partial r^2}\,\mathrm{T}(r))+2\,\frac{\frac{\partial}{\partial r}\,\mathrm{T}(r)}{r}=0$$

```
> sol := simplify(dsolve({deq,inicon}, T(r)));
```

$$sol := \mathrm{T}(r)=\frac{r\,T2\,r2-r\,T1\,r1+r1\,r2\,T1-r1\,r2\,T2}{(-r1+r2)\,r}$$

对给定的参数 r_1, T_1, r_2, T_2, 通过定义球体内的温度分布 $T=T(r)$, 用 Fourier 方法我们得到问题的解析解.

```
> T := unapply(rhs(sol), r);
```

$$T := r \to \frac{r\,T2\,r2-r\,T1\,r1+r1\,r2\,T1-r1\,r2\,T2}{(-r1+r2)\,r}$$

它与用第一种方法得到的解是一样的. 为了得到热流 $\dot{Q}$, 我们用 (14.4)

```
> Q := simplify(-4*Pi*r^2*k*diff(T(r, r1, T1, r2, T2),r));
```

$$Q := -4\,\frac{\pi\,k\,r2\,r1\,(-T1+T2)}{-r1+r2}$$

再次得到与第一种方法一样的结果.

14.2.3 MAPLE 绘图

现在对给定的球壳的热导系数 k, 内外半径 r_1 和 r_2, 以及温度 T_1 和三个不同的 T_2 值，我们作图表示温度的分布 $T=T(r;r_1,T_1,r_2,T_2)$, 画出热流 $\dot{Q}$. 令:

$$k=12[Wm^{-1}K^{-1}],\qquad T_1=400[K],\qquad T_2=300,200,100[K],\qquad r_1=0.1[m],\qquad r_2=0.4[m].$$

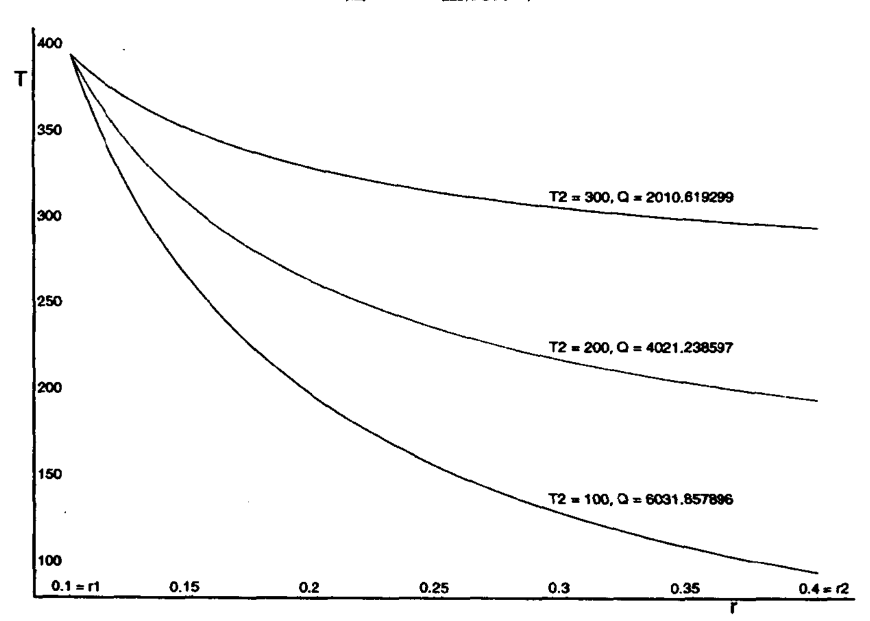
我们利用在 14.2.1 节中给出的表达式计算一般的解析解. 我们将取定 k, T_1, r_1 和 r_2 的值，并在一个循环中计算出三条曲线，最后在一个图中绘出三条曲线. 下面是进行这项工作的完整的程序:

```
> r1 := 0.1:  T1 := 400:  r2 := 0.4:  k := 12:
> TTs := [100, 200, 300]:
```

```
> for T2 in TTs do;
>     Tpl[T2] := T(r);
>     Qpl[T2] := evalf(Q);
> od:
> plot({seq(Tpl[T2], T2 = TTs)}, r = r1..r2);
```



图 14.2 温度分布



14.3 在田野上的非定态热流

考虑一个被太阳晒得变热的田野. 假设田野上初始的温度分布由 T_f 给定, 并且田野表面的温度是常数 T_s(见图 14.3). 设 $k>0$ 是田野的热导系数. 将被确定的非定态温度分布是田野深度 x 和时间 t 的函数.

设 x, y, z 是欧氏空间的坐标. 令原点在田野表面, x 轴正向指向田野内部. 于是温度 $T=T(x,y,z,t)$ 是田野内部点 (x,y,z) 和时间 t 的函数. 由对称性和边界条件, 对所有在平面 $x=c$, $0\le c\le d_{max}$ 上的点 (x,y,z), $T(x,y,z,t)=const(t)$, 其中 d_{max} 是田野的最大深度. 因此, 我们可将 $T=T(x,t)$ 看作两个变量 x 和 t 的函数. 众所周知, 温度分布 $T=T(x,t)$ 是热方程的解 [1].

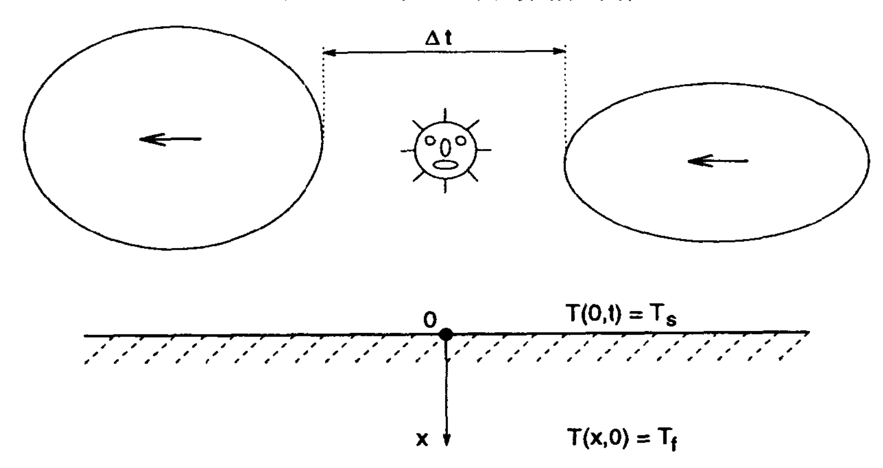
$$a^2\frac{\partial^2 T}{\partial x^2}=\frac{\partial T}{\partial t} \tag{14.12}$$

其中记 $a^2=k$.

我们要求 (14.12) 形如 $V(u)$ 的单个变量 u 的解,

$$V(u)=T(x,t) \tag{14.13}$$

图 14.3 田野的边界和初始化条件



其中

$$u = u(x,t) = \frac{x}{2a\sqrt{t}}. \tag{14.14}$$

方程 (14.14) 用一个变量 u 代替两个变量 x 和 t. 这个替换将偏微分方程 (14.12) 转为一个二阶常微分方程. 这个想法可在 MAPLE 中实现:

```
> restart;
> Subs := u = x/(2*a*sqrt(t)):
> T(x,t):=V(rhs(Subs));
```

$$\mathrm{T}(x,\,t) := \mathrm{V}(\frac{1}{2}\,\frac{x}{a\,\sqrt{t}})$$

```
> eq := a^2*diff(T(x,t), x, x) = diff(T(x, t), t);
```

$$eq := \frac{1}{4}\,\frac{(D^{(2)})(V)(\frac{1}{2}\,\frac{x}{a\,\sqrt{t}})}{t} = -\frac{1}{4}\,\frac{\mathrm{D}(V)(\frac{1}{2}\,\frac{x}{a\,\sqrt{t}})\,x}{a\,t^{3/2}}$$

```
> eq := subs(x = solve(Subs, x), eq)*t*4;
```

$$eq := (D^{(2)})(V)(u) = -2\,\mathrm{D}(V)(u)\,u$$

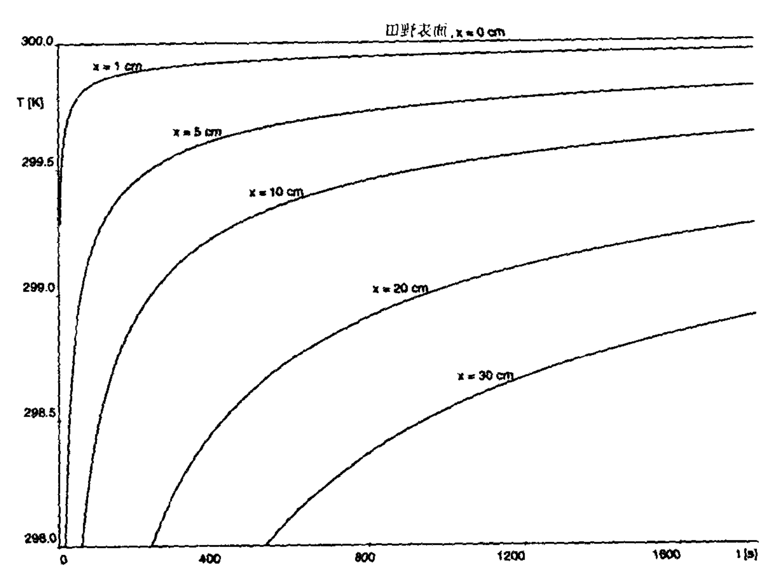
这样我们获得了一个二阶常微分方程

$$\frac{\mathrm{d}^2V(u)}{\mathrm{d}u^2} = -2u\frac{\mathrm{d}V(u)}{\mathrm{d}u}, \tag{14.15}$$

利用 $\frac{\mathrm{d}V(u)}{\mathrm{d}u} = W$ 可手工将其解出.

$$\begin{array}{lllll}
 & \frac{\mathrm{d}^2V(u)}{\mathrm{d}u^2} = -2u\frac{\mathrm{d}V(u)}{\mathrm{d}u} & \Longrightarrow & \frac{\mathrm{d}W}{\mathrm{d}u} = -2uW \\
\Longrightarrow & \frac{\mathrm{d}W}{W} = -2u\mathrm{d}u & \Longrightarrow & \text{可以积分} \\
\Longrightarrow & ln|W| = -u^2 + C_1' & \Longrightarrow & \frac{\mathrm{d}V(u)}{\mathrm{d}u} = C_1\mathrm{e}^{-u^2} \\
\Longrightarrow & \mathrm{d}V(u) = C_1\mathrm{e}^{-u^2}\mathrm{d}u & \Longrightarrow & V(u) = C_1\int_0^u \mathrm{e}^{-p^2}\mathrm{d}p + C_2.
\end{array}$$

图 14.4 深度为 0.0,0.01,0.05,0.1,0.2 和 0.3[m] 的温度场



由 MAPLE 得到的 (14.15) 的解是

```
> Sol := dsolve(eq, V(u));
```

$$Sol := \mathrm{V}(u) = _C1 + _C2\,\mathrm{erf}(u)$$

正如我们所看到的, MAPLE 能够解这个方程. 两个解是相同的, 因为 `erf(u)` (Gauss 或误差函数) 被定义为

$$erf(u) = \frac{2}{\sqrt{\pi}} \int_0^u \mathrm{e}^{-p^2}\,\mathrm{d}p. \tag{14.16}$$

现在将 x 和 t 代回到解中:

```
> T := unapply(rhs(subs(u=rhs(Subs),Sol)), x, t);
```

$$T := (x, t) \to _C1 + _C2\,\mathrm{erf}(\frac{1}{2}\,\frac{x}{a\,\sqrt{t}})$$

用初值条件和边界条件 (见图 14.3) 确定常数 _C1 和 _C2. 我们规定边界条件为 $T(0,t) = \lim_{x\to 0_+} T(x,t) = T_s$, 初始条件为 $T(x,0) = \lim_{t\to 0_+} T(x,t) = T_f$, 即当 $t = 0$ 时整个田野的温度是常数 T_f. 在 MAPLE 中我们假设 $a \geq 0$ 和 $x \geq 0$.

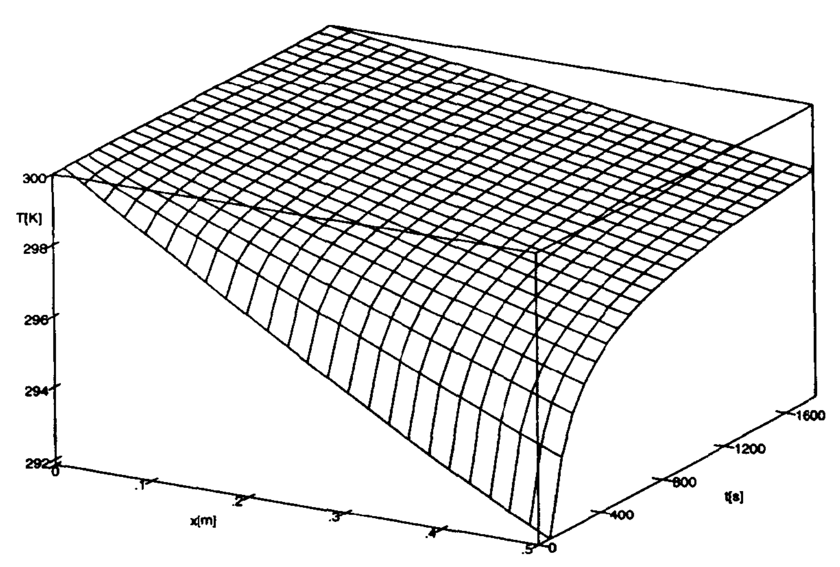
```
> assume(a >= 0): assume(x >= 0):
> BounCon := Ts = limit(T(x,t), x = 0, right);
```

$$BounCon := Ts = _C1$$

```
> IniCon := Tf = limit(T(x, t), t = 0, right);
```

$$IniCon := Tf = _C1 + _C2$$

图 14.5 温度场作为深度和时间的函数




```
> Sol := solve({IniCon, BounCon}, {_C1, _C2});
```

$$Sol := \{_C1 = Ts,\ _C2 = Tf - Ts\}$$

```
> assign(Sol);
> T(x,t);
```

$$Ts + (Tf - Ts)\operatorname{erf}\left(\frac{1}{2}\,\frac{x^{\sim}}{a^{\sim}\sqrt{t}}\right)$$

```
> evalb(diff(T(x,t),x,x) - diff(T(x,t), t)/a^2=0);
```

$$true$$

重新绘制后，获得最后的形式

$$T(x,t) = \frac{2(T_f - T_s)}{\sqrt{\pi}} \int_0^{\frac{x}{2a\sqrt{t}}} \mathrm{e}^{-p^2}\mathrm{d}p + T_s.$$

14.3.1 MAPLE 绘图

让我们计算并绘制出田野内的温度分布，它的最大深度为 $d_{max} = 0.5[m]$, 热传导系数为 $k = a^2 = 0.003[m^2 s^{-1}]$. 太阳照耀了半小时 $\Delta t = 1800[s]$, 在这段时间内田野的表面温度变为 $T_s = 300[K]$. 田野的最初温度为 $T_f = 285[K]$.

```
> Ts := 300:  Tf := 285:  a := sqrt(0.003):
> DVEC := [0, 0.01, 0.05, 0.1, 0.2, 0.3]:
> for d in DVEC do Td[d] := T(d, t) od:
> plot({seq(Td[d], d = DVEC)}, t = 0..1800, 298..300);
> plot3d(T(x, t), x = 0..0.5, t = 0..1800,
>         axes = boxed, orientation = [-50, 50]);
```

参考文献

[1] D. G. ZILL AND M. R. CULLEN, *Advanced Engineering Mathematics*, PWSKENT, Boston, 1992.

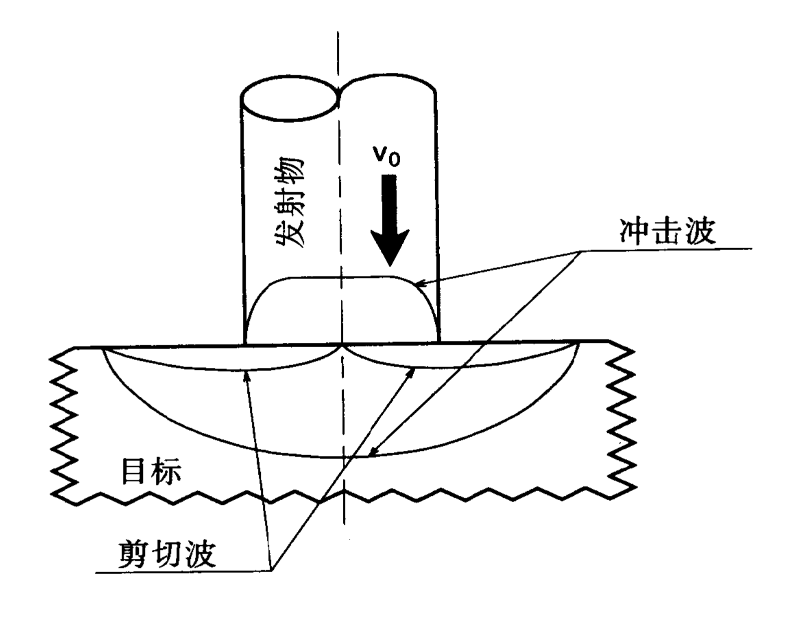
第十五章　模拟贯穿现象

J.Buchar and J.Hřebíček

15.1　引言

贯穿现象在许多领域内都是令人感兴趣的课题 (参见 [3]). 它们常常与核废料的保存, 保护太空船或卫星免遭碎片和陨石撞击等问题有关. 贯穿 (penetration) 的正式定义为发射物穿入目标但非全部穿过. 贯穿现象可以由撞击的角度, 目标和发射物的几何与材料的特征, 以及撞击速度描述. 在这一章我们只考虑通常发生的一根长棒对一个半无界目标的撞击. 这个模型可以描述, 例如, 一个很厚的铁甲被一个高能发射物贯穿. 解决这个问题的最有效的办法是用有限元方法作数值模拟. 许多有限元计算程序能够处理非常复杂的材料结构关系. 这些程序是很昂贵的, 且需要大量的运行时间. 这也就是为什么简单的一维理论仍然具有相当价值的主要原因. 这样的理论也有助于了解物理参数和它们有关的结果之间的相互关系. 这些相互关系通常难以从上面提到的计算分析中确定. 因此, 简单的理论常常为实验的设计提供了基础, 使得有限元方法的数值模拟成为可行. 按这个设想, 我们将研究某些用 MAPLE 处理的贯穿模型.

图 15.1　瞬时冲击状态的波形



15.2　贯穿理论的简短描述

研究贯穿现象的文献很多, 尽管它的第一个模型是 60 年代才建立的 [8]. 对一个半无界目标的贯穿模型由下面四条规则构成:

1. 瞬时冲击规则：在初始接触时，冲击波在目标和发射物中产生，图象显示在图 15.1 中．材料的反应主要由它的可压缩性和密度确定．冲击压力通常很大，足以引起广泛的塑性流，溶化和蒸发．
2. 定态规则：在贯穿期间内发射物腐蚀，同时形成空穴．这个阶段通常由一个改进的 Bernoulli 方程描述．
3. 空穴规则：在发射物完全耗竭后，空穴继续扩大，这是由于它自身的惯性和在目标压力下材料强度减小时获得的能量．强度的减小是多次波反射的结果 [9]．
4. 恢复规则：空穴尺度的略微减小是由于弹性的反弹作用．这种反弹也许会产生高度的张力压强而引起断裂，正如对恢复的目标常常能观察到的那样 [5]．

这个模型的主要特征已靠计算证实 [11]．数值模拟导致了对贯穿过程的最复杂的分析．模拟是十分昂贵的，因为它需要花许多小时的 CPU 时间．这个程序即不便于计算一系列的冲击实验，也不便于深入研究在介绍中提到的贯穿特征．这些事实强烈地激励发展贯穿过程的新解析模型．关于贯穿有许多解析模型，见综述 [2]．应用最广泛的是一维模型，称为 Tate 模型 [10]，虽然它也曾由 Alekseevskii[1] 独立地提出．这个模型被看作是厚目标被长棒贯穿的标准模型，它的主要特征在下面几节阐述．

15.3 Tate-Alekseevskii 模型

这个模型假设发射物是坚硬的，只在发射物与目标接触的很薄区域内发生腐蚀．这个区域没有空间尺度，只包含目标与发射物的表面．这个表面的变化由一个改进的 Bernoulli 方程描述

$$\frac{1}{2}\rho_p(v-u)^2+Y_p=\frac{1}{2}\rho_t u^2+R_t \tag{15.1}$$

其中 ρ_p 和 ρ_t 分别是发射物和目标的密度， v 是发射物尾部的速度， u 是贯穿的速度． Y_p 是发射物的强度， R_t 是一维形式的目标阻力．

穿入的发射物被沿着发射物长度方向接触的表面传递来的力阻挡而减速．减速度由

$$\rho_p l\frac{\mathrm{d}v}{\mathrm{d}t}=-Y_p \tag{15.2}$$

给出，其中 $l=l(t)$ 是发射物的现时长度．由于腐蚀作用，棒的长度变化由棒的未形变部分的运动牛顿方程给出

$$\frac{\mathrm{d}l}{\mathrm{d}t}=-(v-u). \tag{15.3}$$

方程 (15.1)-(15.3) 是关于未知函数 $l=l(t)$, $v=v(t)$ 和 $u=u(t)$ 的方程组，带初始条件

$$v(0)=v_0,\qquad l(0)=L, \tag{15.4}$$

其中 v_0 是冲击速度， L 是发射物的初始长度．贯穿速度 u 可以从方程 (15.1) 获得：

```
>     # Bernoulli equation
> eq := (1/2)*rho[p]*(v-u)^2+Yp=(1/2)*rho[t]*u^2+Rt:
> pen := solve(eq, u);
```

$$pen := \frac{1}{2}\,\frac{2\,\rho_p\,v+2\,\sqrt{2\,\rho_p\,Rt-2\,\rho_p\,Yp+\rho_t\,\rho_p\,v^2-2\,\rho_t\,Rt+2\,\rho_t\,Yp}}{\rho_p-\rho_t},$$
$$\frac{1}{2}\,\frac{2\,\rho_p\,v-2\,\sqrt{2\,\rho_p\,Rt-2\,\rho_p\,Yp+\rho_t\,\rho_p\,v^2-2\,\rho_t\,Rt+2\,\rho_t\,Yp}}{\rho_p-\rho_t}$$

```
>         # We take the second solution
>         # because of u < v and simplify it
>  u_v[1] := simplify(pen[2]):
```

对具相同密度的发射物和目标，即 $\rho_p = \rho_t$, u 值可如下计算

```
>  simplify(series(u_v[1],rho[p]=rho[t]),symbolic):
>  u_v[2] := subs({rho[t]=rho,rho[p]=rho},op(1,"));
```

$$u_v_2 := \frac{1}{2}\,\frac{\rho v^2 - 2\,Rt + 2\,Yp}{\rho\,v}$$

于是，对 $\rho_p \neq \rho_t$, 贯穿速度 u 为

$$u = \frac{\rho_p v - \sqrt{2(R_t - Y_p)(\rho_p - \rho_t) + \rho_p \rho_t v^2}}{\rho_p - \rho_t} \tag{15.5}$$

而对 $\rho_p = \rho_t$,

$$u = \frac{\rho v^2 + 2(Y_p - R_t)}{2\rho v}. \tag{15.6}$$

将 (15.2) 代入 (15.3) 得

$$\frac{\mathrm{d}l}{\mathrm{d}v} = \frac{l(v)\rho_p(v-u)}{Y_p}. \tag{15.7}$$

用 MAPLE 求微分方程 (15.7) 具初值 $l(v_0) = L$ 的解

```
>      # Definition of initial value problem
>  deq := diff(l(v), v)=l(v)*rho[p]*(v-u(v))/Yp:
>  incond := l(v0) = L:
>  len[1] := simplify (rhs (dsolve({deq, incond}, l(v))));
```

$$len_1 := e^{\left(\frac{\rho_p \int_{v0}^{v} u - \mathrm{u}(u)\, du}{Y_p}\right)} L$$

得到发射物现有长度 l, 当发射物密度与目标密度相同，即 $\rho_p = \rho_t$ 时，此式可简化:

```
>  subs(u=unapply(u_v[2],v),{rho[p]=rho,rho[t]= rho},len[1]):
>  len[2] := simplify (expand ("));
```

$$len_2 := e^{\left(1/4\,\frac{\rho\,(v-v0)\,(v+v0)}{Yp}\right)}\, v^{\left(-\frac{-Rt+Yp}{Yp}\right)}\, v0^{\left(\frac{-Rt+Yp}{Yp}\right)}\, L$$

在时刻 T 棒的贯穿深度 pe 为

$$pe = \int_0^T u(t)\mathrm{d}t. \tag{15.8}$$

由 (15.2) 得

$$pe = \frac{\rho_p}{Y_p}\int_{v(T)}^{v_0} ul\mathrm{d}v. \tag{15.9}$$

将关于 $u(v)$ 和 $l(v)$ 的式子代入方程 (15.9), 我们得到通常的贯穿方程

$$\frac{pe}{L} = F(v_0, R_t, Y_p, \rho_t, \rho_p). \tag{15.10}$$

一般上面的积分不能被解析地求解，但我们将在下一节对特别的参数值给出 (15.10) 解的精确解.

15.3.1 特殊情况 $R_t = Y_p$

对相同的目标阻力 R_t 和发射物的强度 Y_p, 我们计算 $\lim_{T\to\infty} p(T)$. 显然 $T\to\infty$ 对应 $v\to 0$.

```
> u_v[3] := simplify (subs(Rt=Yp, u_v[1]), symbolic):
> subs(u=unapply(u_v[3],v),len[1]):
> l := simplify(", symbolic):
> pe := simplify(int(l*u_v[3], v=0..v0)):
```

标准化的贯穿深度 pe/L 为

$$\frac{pe}{L} = \frac{Y_p}{\rho_p}\frac{(\rho_p-\sqrt{\rho_p\rho_t})}{(\rho_t-\sqrt{\rho_p\rho_t})}\left(-1+\mathrm{e}^{\frac{\rho_p(\rho_t-\sqrt{\rho_p\rho_t})v_0^2}{2(\rho_p-\rho_t)Y_p}}\right). \tag{15.11}$$

对 $\rho_p=\rho_t=\rho$, 方程 (15.11) 可被简化

```
> simplify(limit(pe,rho[t]=rho[p]),symbolic):
```

这样我们得到

$$\frac{pe}{L} = \frac{Y_p}{\rho}(1-e^{-\frac{\rho v_0^2}{4Y_p}}). \tag{15.12}$$

15.3.2 特殊情况 $\rho_p=\rho_t=\rho$

对相同的贯穿物密度和目标密度，我们分下面几种情况.

情况 1: $R_t\geq Y_p$. 根据 (15.1), 仅当

$$\frac{1}{2}\rho_p v_0^2 + Y_p \geq R_t \tag{15.13}$$

时贯穿发生. 当 $R_t = mY_p$ 时，速度 v 在 v_0 与 v_m 之间，其中

$$v_m = \sqrt{\frac{2(m-1)Y_p}{\rho_p}}. \tag{15.14}$$

标准化的贯穿深度可如下确定:

```
> vm := sqrt(2*(m-1)*Yp/rho):
> simplify (subs(Rt=m*Yp,len[2]*u_v[2])):
> pe := (rho/Yp)*int(", v=vm..v0):
```

它可被整齐地写作

$$\frac{pe}{L} = \frac{v_0 L}{2Y_p}\int_{v_m}^{v_0}\mathrm{e}^{-\frac{\rho(v_0-v)(v_0+v)}{4Y_p}}(\rho v^2+2Y_p-2mY_p)(\frac{v}{v_0})^m v^{-2}\mathrm{d}v.$$

对 $m\in\mathbf{N}$, 上面表达式可以进一步简化，例如 p_1 和 p_3:

```
> pe1 := simplify(eval(subs(m=1, pe)));
```

$$pe1 := -L\left(-1+e^{(-1/4\,\frac{\rho\,v0^2}{Yp})}\right)$$

```
> pe3 := simplify(eval(subs(m=3, pe)));
```

$$pe3 := \frac{L\left(v0^2\,\rho - 8\,Yp + 4\,Yp\,e^{(1/4\,\frac{-v0^2\,\rho+4\,Yp}{Yp})}\right)}{\rho\,v0^2}$$

情况 2: $R_t < Y_p$. 由不等式

$$Y_p \geq \frac{1}{2}\rho_t u^2 + R_t, \qquad 当 \qquad u = v_0, \tag{15.15}$$

$$Y_p < \frac{1}{2}\rho_t u^2 + R_t, \qquad 当 \qquad u < v_0 \tag{15.16}$$

描述了两个不同的贯穿过程的速度规则. 如果 (15.15) 成立, 棒象一个坚硬物体以撞击速度 v_0 贯穿. 对极限速度

$$v_l = \sqrt{\frac{2(Y_p - R_t)}{\rho_t}}, \tag{15.17}$$

(15.15) 中等式成立. 对 $v_0 \leq v_l$, 在棒尖端的减速压力为 $\tilde{P} = \frac{1}{2}\rho_t v_0^2 + R_t$. 将压力 $\tilde{P}$ 代入方程 (15.2) 我们得到

$$\rho_p L \frac{\mathrm{d}v}{\mathrm{d}t} = -(\frac{1}{2}\rho_t v_0^2 + R_t). \tag{15.18}$$

在 v_0 与零速度之间积分得

$$p_r = L(\frac{\rho_p}{\rho_t})^2 \ln(1 + \frac{\rho_t v_0^2}{2R_t}). \tag{15.19}$$

对 $v_0 > v_l$, 通过对 (15.9) 从 v_0 到 v_l 的积分得到贯穿深度 p_f. 总的贯穿为

$$pe = p_r + p_f. \tag{15.20}$$

贯穿理论的基本方程 (15.1)-(15.3) 是非线性的. 通常它们的解只能由数值计算. MAPLE 使我们能够获得一个级数形式的逼近解. 在 [4] 中的详细分析表明, 用一个 5th 阶多项式 u_{appr} 逼近贯穿速度 u 就足够了.

```
> restart:
>   # Bernoulli equation for same densities rho=rho[t]=rho[p]
> eqe := (1/2)*rho*(v(t)-u)^2+Yp=(1/2)*rho*u^2+Rt:
> u := solve(eqe,u):
> deqe1 := rho*l(t)*diff(v(t), t)=-Yp:
> deqe2 := diff(l(t), t)=-(v(t)-u):
> incond := l(0)=L, v(0)=v0:
> sol := dsolve({deqe1, deqe2, incond}, {l(t), v(t)}, series):
>   # Polynomial approximation of v(t) and l(t)
> v := op(2,op(select(x->op(1,x)='v(t)',sol))):
> l := op(2,op(select(x->op(1,x)='l(t)',sol))):
>   # Polynomial approximation of u(t)
> uappr := convert(series(v/2+(Yp-Rt)/(rho*v),t=0,6),polynom):
>   # First few coefficients of uappr
> series(uappr,t=0,2);
```

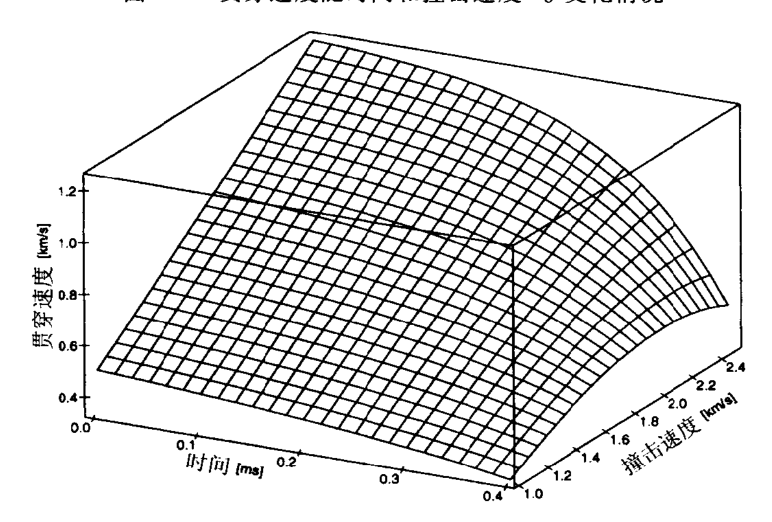
$$(\frac{1}{2}\,v0 + \frac{Yp - Rt}{\rho\, v0}) + (-\frac{1}{2}\,\frac{Yp}{\rho\, L} - \frac{(-Yp + Rt)\; Yp}{\rho^2\, v0^2\, L})\,t + \mathrm{O}(t^2)$$

我们将这个模型用于下面的例子.

```
> rho := 7810:  # Target and penetrator density
> L := 0.25:  # Length of rod
>   # Target resistance Rt = 900*10^6
>   # Strength of projectile Yp = 900*10^6
> u := subs(Rt=900*10^6,Yp=900*10^6,uappr):
> plot3d(u,t=0..0.0004,v0=1000..2500,
>        axes=boxed,orientation=[-30,60]);
```

其中参数取自射击 No.B19, 见 [4]: $\rho = \rho_t = \rho_p = 7810[kgm^{-3}]$, $L = 0.25[m]$, $R_t = Y_p = 9.10^8[Nm^{-2}]$. 计算结果 (展示在图 15.2 中) 与用有限元程序 AUTODYN 2D 得到的二维数值模拟的结果进行了比较. 在 [4] 中证实这些结果基本一致. 下面的贯穿模型考虑了在棒顶部的腐蚀过程.

图 15.2 贯穿速度随时间和撞击速度 v_0 变化情况



15.4 腐蚀棒的贯穿模型

假设一根初始截面积为 A_p, 直径为 d_p, 长度为 l_p 的圆柱棒以速度 v_0 撞击半无界的目标 (见图 15.3). 目标由强度为 Y_t, 密度为 ρ_t 的材料制成, 棒的强度为 Y_p, 密度为 ρ_p. 实验表明在贯穿期间棒被腐蚀. 这个腐蚀过程出现在棒的顶端, 使得棒的初始质量 $m_p = \rho_p A_p l_p$ 减小, 还减小了棒的长度 l_p. 单位时间内失去的棒质量为

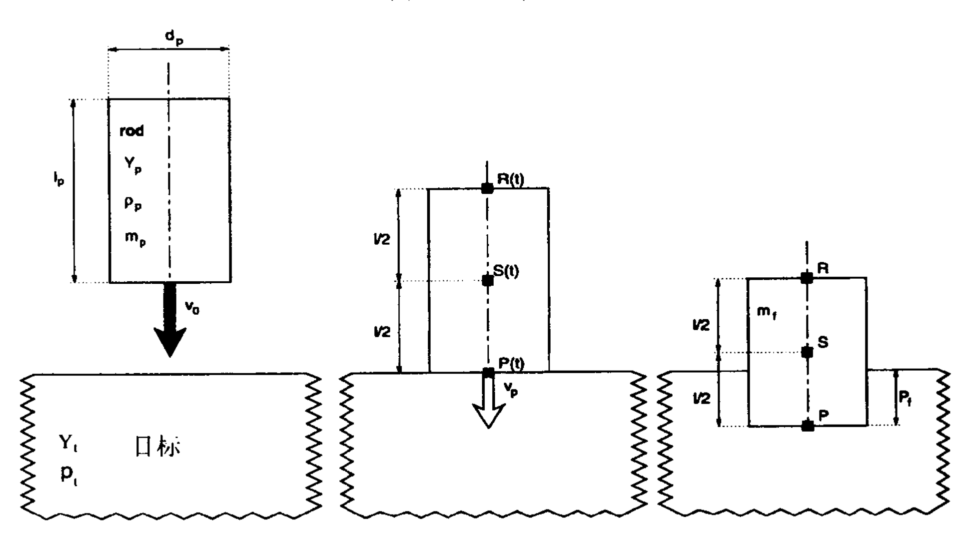
$$\dot{m} = \rho_p A_p \dot{l},$$

其中 $m(t)$ 和 $l(t)$ 是棒的瞬时质量和长度, 点表示对时间 t 的微分. 因 $\dot{m} < 0$, 用符号 $\dot{m}_p = |\dot{m}|$ 是方便的, 即

$$\dot{l} = -\frac{\dot{m}_p}{\rho_p A_p}. \tag{15.21}$$

设 $S(t)$ 是在时刻 t 棒中心的位置, 即棒的顶部 $P(t)$ 与尾部 $R(t)$ 之间的中点. 我们忽略棒撞击时

图 15.3 贯穿过程



张力的传播. 棒顶部和尾部的速度为

$$\dot{P} = \dot{S} + \frac{\dot{l}}{2} = \dot{S} - \frac{\dot{m}_p}{2\rho_p A_p}, \tag{15.22}$$

$$\dot{R} = \dot{S} - \frac{\dot{l}}{2} = \dot{S} + \frac{\dot{m}_p}{2\rho_p A_p}. \tag{15.23}$$

考虑棒顶部的情况. 质量 Δm_p 被丢失, 而棒中心 (质量的中心) 的速度变为 $\dot{S} + \Delta\dot{S}$. 动量守恒为

$$[(m - \Delta m_p)(\dot{S} + \Delta\dot{S}) + \dot{P}\Delta m_p] - m\dot{S} = -F\Delta t, \tag{15.24}$$

其中 F 是由目标阻碍棒的运动产生的力. 对 $\Delta t \to 0$ 取极限, 由方程 (15.24) 得

$$m\frac{\mathrm{d}\dot{S}}{\mathrm{d}t} = (\dot{S} - \dot{P})\dot{m}_p - F. \tag{15.25}$$

将方程 (15.22) 代入 (15.25) 我们得到

$$m\frac{\mathrm{d}\dot{S}}{\mathrm{d}t} = a_p\dot{m}_p^2 - F, \text{ 其中记} a_p = \frac{1}{2\rho_p A_p}.$$

在 [7] 中证明了 F 可以表达成

$$F = a + b\dot{P}^2. \tag{15.26}$$

常数 $a = 3Y_t A_p$, 依赖目标强度. 这表明目标在受发射物撞击时立即抵抗, 发射物顶部暂停了一定的时间 [6]. 第二项 b 是单位体积的动能或单位面积的力, 即 $b = \rho_t A_p/2$.

棒的腐蚀可以描述为 (见 [4]),

$$\dot{m} = -\mu_0\dot{P} \tag{15.27}$$

或

$$\dot{m}_p = \mu_p\dot{P}, \tag{15.28}$$

其中 μ_p 为未知参数. 因为在 $t=0$ 时贯穿的深度 $P(t)$ 为零, 即 $P(0)=0$, 微分方程 (15.27) 的解为

$$m(t) = m_p - \mu_p P(t). \tag{15.29}$$

设 P_f 是棒顶部的最终贯穿深度, m_f 是在 t_f 贯穿完成时棒的剩余质量. 参数 μ_p 取值

$$\mu_p = \frac{m_p - m_f}{P_f}, \tag{15.30}$$

因此 μ_p 的最大值为 $\mu_{max} = m_p/P_f$. 将方程 (15.28) 代入 (15.22), 我们得

$$\dot{S} = (1+\mu_p a_p)\dot{P} = k\dot{P},\ 其中 k = k(\mu_p) = 1 + \mu_p a_p. \tag{15.31}$$

利用方程 (15.26), (15.28), (15.29) 和 (15.31), 方程 (15.25) 可改写成

$$(m_p - \mu_p P)k\frac{\mathrm{d}\dot{P}}{\mathrm{d}t} = -(a + c\dot{P}^2), \quad 其中\ \ c = c(\mu_p) = b - a_p\mu_p^2. \tag{15.32}$$

微分方程 (15.32) 关于初始条件

$$P(0) = 0, \tag{15.33}$$

$$\dot{P}(0) = v_p, \tag{15.34}$$

的解给出了贯穿深度对时间的依赖关系 $P = P(t)$. 根据 Tate 理论 [6], 在 (15.34) 中的初始贯穿速度 v_p 被确定为

$$v_p = \frac{v_0}{2} + \frac{Y_t - Y_p}{\rho_t v_0}. \tag{15.35}$$

因为我们不能由实验测量 $P(t)$, 所以必须寻找方程 (15.32) 的某些特征. 如果我们考虑到

$$\frac{\mathrm{d}\dot{P}}{\mathrm{d}t} = \frac{\mathrm{d}\dot{P}}{\mathrm{d}P}\frac{\mathrm{d}P}{\mathrm{d}t}$$

则方程 (15.32) 可转化为

$$-\frac{k}{a + c\dot{P}^2}\dot{P}\frac{\mathrm{d}\dot{P}}{\mathrm{d}P} = \frac{1}{m_p - \mu_p P},$$

带初始条件 (15.34).

用 MAPLE 可以解上面的初值问题.

```
> restart;
> de := -k*DP(P)*diff(DP(P), P)/(a + c*DP(P)^2) =1/(mp - mu*P);
```

$$de := -\frac{k\,\mathrm{DP}(P)\,(\frac{\partial}{\partial P}\,\mathrm{DP}(P))}{a + c\,\mathrm{DP}(P)^2} = \frac{1}{mp - \mu\,P}$$

```
> ic := DP(0) = vp:
> pen := dsolve({de,ic},DP(P));
```

$$pen := \mathrm{DP}(P) = \frac{\sqrt{c\left(-a + \frac{e^{(2\,\frac{\ln(mp-\mu\,P)\,c}{\mu\,k})}\,(a + vp^2\,c)}{(mp^{(\frac{c}{\mu\,k})})^2}\right)}}{c},$$

$$\mathrm{DP}(P) = -\frac{\sqrt{c\left(-a + \frac{e^{(2\,\frac{\ln(mp-\mu\,P)\,c}{\mu\,k})}\,(a + vp^2\,c)}{(mp^{(\frac{c}{\mu\,k})})^2}\right)}}{c}$$

```
> sol := op(2,pen[2]);
```

$$sol := -\frac{\sqrt{c\left(-a+\frac{e^{(2\frac{\ln(mp-\mu P)c}{\mu k})}(a+vp^2c)}{(mp^{(\frac{c}{\mu k})})^2}\right)}}{c}$$

```
> sol1 := subs(2*c/(mu*k)=1/e,sol);
```

$$sol1 := -\frac{\sqrt{c\left(-a+\frac{e^{(2\frac{\ln(mp-\mu P)c}{\mu k})}(a+vp^2c)}{(mp^{(\frac{c}{\mu k})})^2}\right)}}{c}$$

对解做一些简单处理，我们得到

$$\dot{P}=\sqrt{\frac{a}{c}\left((1+\frac{c}{a}v_p^2)(1-\frac{\mu_p}{m_p}P)^{1/\varepsilon}-1\right)}, \tag{15.36}$$

其中

$$\varepsilon=\varepsilon(\mu_p)=\frac{\mu_p k}{2c}$$

是较小的腐蚀率.

当射入一个半无界目标时，在 t_f 时刻 $\dot{P}=0$, 由 (15.36) 得

$$P_f=\frac{m_p}{\mu_p}\left(1-(1+\frac{c(\mu_p)}{a}v_p^2)^{-\varepsilon(\mu_p)}\right). \tag{15.37}$$

对参数 μ_p 的计算，方程 (15.37) 比 (15.30) 更适用，因为难以确定剩余棒的质量 m_f. 而在每次最后的发射实验中数据 $[P_f, v_0]$ 是可以得到的.

模拟发射实验包含如下步骤:

1. 确定参数 μ_p. 如果在这一章提供的贯穿理论正确，则对每组棒和目标的材料组合 μ_p 应该是常数，不依赖撞击速度 v_0. 我们将用 MAPLE 解方程 (15.37) 求 μ_p 的数值解，并估计当 $\mu_p\to 0$ 时 (15.37) 的极限. 这个极限对应于一个硬发射物的贯穿. 我们将定义 k, c 和 ε 为变量 μ 的函数. P_f 也被定义为 μ 的函数. 我们将方程 $P_f(\mu)=P_f$ 的解记为 μ_p.
2. 计算速度 $\dot{P}$, 它沿贯穿路径 $P\in[0,P_f]$ 是 P 的函数.
3. 为了求得函数 $P(t)$ 及它的图形，求解初值问题 (15.32), (15.33) 和 (15.34).
4. 研究目标的强度 Y_t 对在 1-3 中确定的参数和函数的影响. 值 Y_t 可能受到张力的影响，它常常高于由传统材料试验所得到的值.

15.5 数值实验

我们将用 MAPLE 解一个具体的问题. 考虑下面 J. Buchar [4] 的射击 No. C06 的实验数据. $m_p=1.491[kg]$, $v_0=1457.9[ms^{-1}]$, $\rho_t=\rho_p=7810[kgm^{-3}]$, $d_p=0.0242[m]$, $Y_t=9.7\cdot10^8[Nm^{-2}]$, $Y_p=15.81\cdot10^8[Nm^{-2}]$, $P_f=0.245[m]$.

我们用 MAPLE 解上面的例子，并进行弹道实验分析的四个步骤. 第一步我们求 μ_p, 然后绘图 15.4, $P_f(t)$ 在区间 $[0,\mu_{max}]$ 上的图. 第二步我们绘出图 15.5, 贯穿速度 $\dot{P}$ 作为 P 的函数的图，并

图 15.4 μ 对贯穿深度的影响

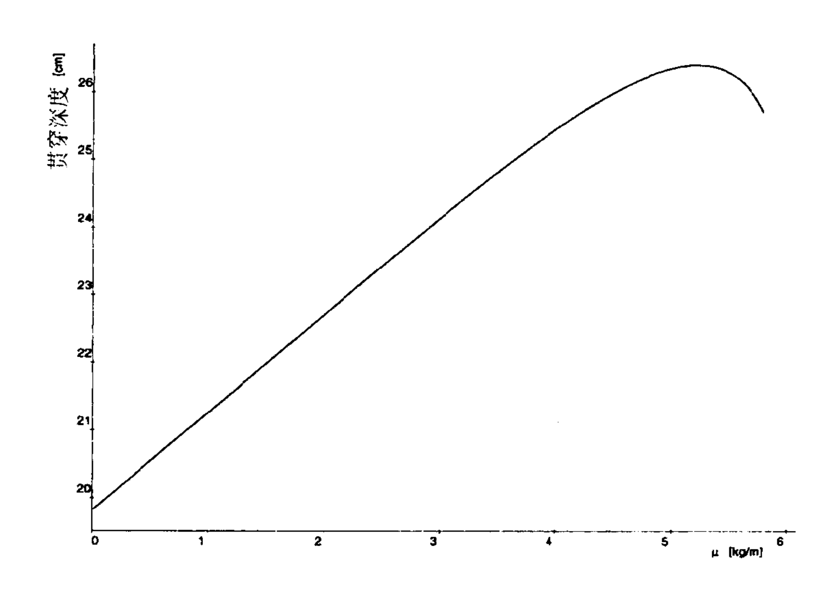


图 15.5 贯穿速度对深度的依赖关系

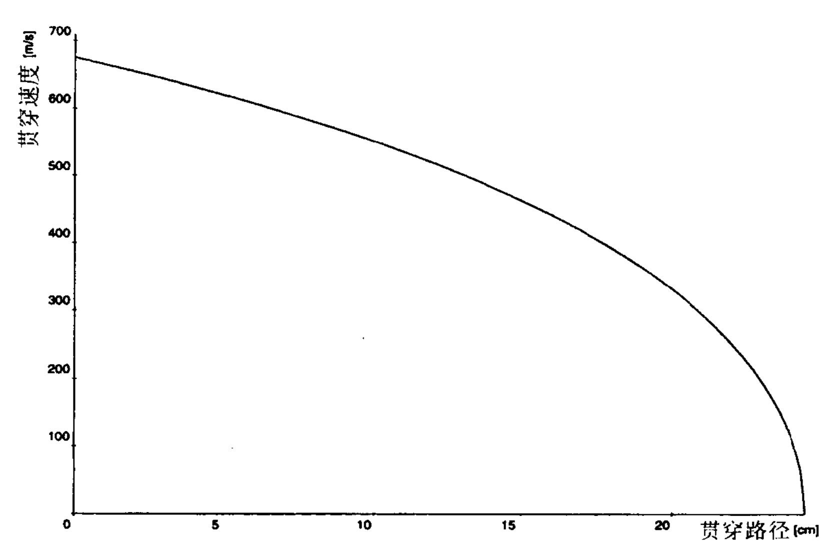
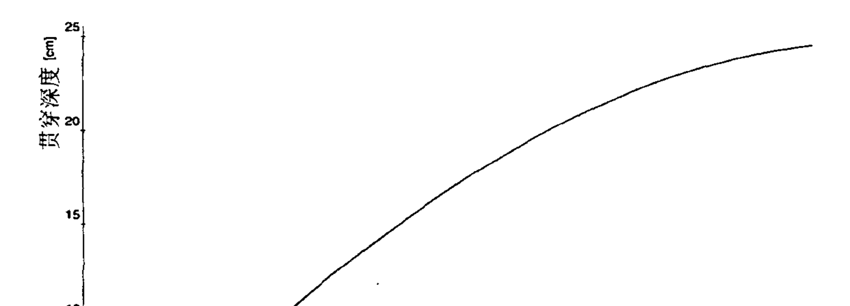


图 15.6 深度对时间的依赖关系



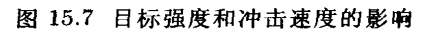
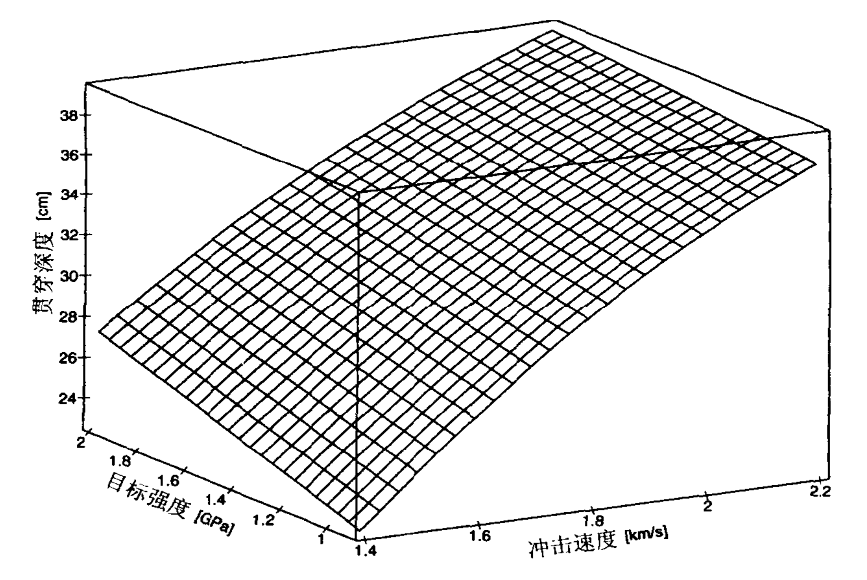


图 15.7 目标强度和冲击速度的影响



确定参数 μ_p, 见 (15.37). 第三步我们将求上面的初值问题的数值解，并绘图 15.6, 解 $P(t)$ 在区间 $(0, t_f)$ 上的图， $t_f = 0.0006$. 第四步我们将绘图 15.7, 贯穿深度作为 Y_t 和 v_0 的函数的图.

```
>     # Projectile and target data and the final penetration depth
>   rho[t] := 7810:  mp := 1.491:  Yt := 9.70*10^8:
>   rho[p] := rho[t]:  dp := 0.0242:  Yp := 15.81*10^8:
>   Ap := Pi*dp^2/4:  v0 := 1457.9:
>   pf := 0.245:
>
>     # 1st step - Determination of mup and graph penetration depth
>     # vs erosion rate mu, the definition of constants ap, a, b
>     # and initial penetration velocity
>  ap := 1/(2*rho[p]*Ap):  a := 3*Yt*Ap:
>  b := rho[t]*Ap/2:  vp := v0/2+(Yt-Yp)/(rho[t]*v0):
>     # The definition of functions k, c, eps and Pf of mu
>  k := mu->1+ap*mu:  c := mu->b-ap*mu^2:
>  eps := mu->mu*k(mu)/(2*c(mu)):
>  Pf := mu->mp*(1-(1+c(mu)*vp^2/a)^(-eps(mu)))/mu:
>     # - Graph of influence of mu on the penetration depth
>  plot(Pf(mu),mu=0..6);
>     # Determination of limit
>  Pfzero := limit(Pf(mu),mu=0):
>     # Solution of the equation Pf(mu)=pf
>  mup := fsolve(Pf(mu)=pf,mu):
>
>     # 2nd step - Graph of the penetration velocity
>     # along the penetration path
>  dP := P -> sqrt(a*((1 + c(mup)*vp^2/a)*
>               (1 - mup*P/mp)^(1/eps(mup)) - 1)/c(mup)):
>  plot(dP(P),P=0..pf);
>
>     # 3rd step - Graph of the time dependence of the penetration
>  tf := 0.0006:
>     # The evaluation of the time dependence
>     # of the penetration and its velocity
>  deq := (mp - mup*p(t))*k(mup)*diff(p(t), t, t) =
>         -(a + c(mup)*diff(p(t),t )^2):
>  incond := p(0) = 0, D(p)(0) = vp:
>  F := dsolve({deq, incond}, p(t), numeric):
>  plots[odeplot](F, [t, p(t)], 0..tf);
>
>     # 4th step - The evaluation of the influence
>     # of target strength and impact velocity
>  vp1 := (vi,y) -> vi/2 + (y - Yp)/(rho[t]*vi):
>  depth := (vi,y) ->
>            mp*(1 - (1 + c(mup)*vp1(vi,y)^2/a)^(-eps(mup)))/mup:
>  opt := orientation=[-145,60], axes=boxed:
```

```
> plot3d(depth(vi,y),vi=1400..2200,y=900*10^6..2000*10^6,opt);
```

15.6 结论

上面所展示的由 MAPLE 表示的符号运算方法对研究贯穿现象是非常有效的. 所给的例子表明 MAPLE 有助于解贯穿问题. 描述的程序使我们能够估算目标和发射物的强度对贯穿深度的影响. 以前这个问题是由数值方法求解的. 用 MAPLE 分析发射物的腐蚀也非常容易, [4] 中阐述了这个方法的其它优点.

参考文献

[1] V. P. ALEKSEEVSKII, *Penetration of a rod into a target at high velocity*, Combustion, explosion and shock waves, 2,1966, pp.63-66.

[2] C. E. ANDERSON JR., S. R. BODNER, *Ballistic impact: the status of analytical and numerical modeling*, Int. J. Impact Engng., 7, 1988, pp.9-35.

[3] Z. BÍLEK AND J. BUCHAR, *The behavior of metals under high rates of strain (in Czesch)*, Academia, Praha, 1984.

[4] J. BUCHAR, M. LAZAR, S. ROLC, *On the penetration of steel targets by long rods*, Acta Techn. CSAV 39, 1994, pp.193-220.

[5] D. R. CHRISTMAN, J. W. GEHRING, *Analysis of high-velocity projectile penetration mechanics*, J.Appl.Phys., 27, 1966, pp.63-68.

[6] J. D. CINNAMON ET AL., *A one-dimensional analysis of rod penetration*, Int. J. Impact Engng., 12, 1992, pp.145-166.

[7] J. DEHN, *A unified theory of penetration*, Int. J.Impact Engng.,5, 1987, pp.239-248.

[8] W. HERRMANN, J. S. WILBECK, *Review of hypervelocity penetration theories*, Int.J.Impact Engng., 5, 1987, pp.307-322.

[9] V. K. LUK, M. J. FORRESTAL, D. E. AMOS, *Dynamical spherical cavity-expansion of strain-hardening materials*, J.Appl.Phys., 58, 1991, pp.1-6.

[10] A. TATE, *A theory of deceleration of long rods after impact*, J.Mech. Phys. Solids 15, 1967, pp.287-399.

[11] J. A. ZUKAS, *High velocity impact dynamics*, Wiley Inter science, New York, 1990.

第十六章　玻色粒子系统的热容量

F. Klvaňa

16.1　引言

本章研究在接近绝对零度的低温条件下具有非零质量的玻色粒子 (例如 He^4) 系统，此时出现一种超流性 (或者也出现电子的超导性).

让我们考虑一个由 N 个无相互作用的、具有非零质量 m 的玻色粒子组成的量子系统 (粒子具有整数自旋). 设系统在容积 V 的范围内，与其环境相平衡并具有绝对温度 T. 那么能量在 $[\epsilon, \epsilon+\mathrm{d}\epsilon]$ 区间的粒子数平均值由下列方程给出 [1,2]

$$\langle n(\epsilon)\rangle \cdot \mathrm{d}\epsilon = \frac{g(\epsilon)\cdot \mathrm{d}\epsilon}{e^{(\epsilon-\mu)/\theta}-1}, \qquad \epsilon \in [0,\infty), \tag{16.1}$$

其中 $\langle n(\epsilon)\rangle$ 是粒子数的能量密度， $g(\epsilon)$ 是态密度，在此情形，它由下列方程

$$g(\epsilon) = g_0 V \frac{4\pi}{h^3} 2m^{3/2}\cdot \epsilon^{1/2} = \lambda \cdot \sqrt{\epsilon}, \tag{16.2}$$

给出，其中 $\theta = kT(k$ 是 Boltzmann 常数) 是统计温度.

量 $\mu = \mu(T,V,N)$ 是化学势，其对热力学参数 (T,V,N) 的依赖关系由归一化条件给出

$$N = \int_0^\infty \langle n(\epsilon)\rangle \mathrm{d}\epsilon = \lambda \int_0^\infty \frac{\epsilon^{1/2}\mathrm{d}\epsilon}{\mathrm{e}^{(\epsilon-\mu)/\theta}-1} = \lambda\cdot\theta^{3/2}\int_0^\infty \frac{x^{1/2}\cdot \mathrm{d}x}{\mathrm{e}^{x-\mu/\theta}-1}. \tag{16.3}$$

对玻色子系统能够证明 $\mu \le 0$ 而且 $T=0$ 时 $\mu=0$. 但由等式 (16.3) 可得，对于 $\theta=\theta_c>0(\theta_c$ 称为临界温度), $\mu=0$. 所以，对 $\theta=\theta_c$

$$N = \lambda\cdot\theta_c^{3/2}\int_0^\infty \frac{x^{1/2}\cdot \mathrm{d}x}{\mathrm{e}^x-1}. \tag{16.4}$$

由黎曼 ζ 函数的定义

$$\zeta(\nu) = \frac{1}{\Gamma(\nu)}\int_0^\infty \frac{x^{\nu-1}\mathrm{d}x}{\mathrm{e}^x-1} \tag{16.5}$$

(16.4) 能够写成形式

$$N = \lambda\cdot\Gamma\left(\frac{3}{2}\right)\cdot\theta_c^{3/2}\cdot\zeta\left(\frac{3}{2}\right) \tag{16.6}$$

对于 $\theta<\theta_c$, 我们不得不对系统使用一个不同的模型. 这时， μ 必须为零，同时，归一化条件具有下述形式

$$N - N' = \lambda\int_0^\infty \frac{\epsilon^{1/2}\mathrm{d}\epsilon}{\mathrm{e}^{\epsilon/\theta}-1} \tag{16.7}$$

其中 N' 是被 "凝结" 在单粒子基态 $(\epsilon=0)$ 的粒子个数. 这些被凝结的粒子具有一种称为超流性的特殊行为. 所以 μ 作为 θ 的函数具有两个不同的解析区域：对于 $\theta\le\theta_c$, $\mu=0$, 及对于 $\theta\ge\theta_c$, μ 是方程 (16.3) 的一个解.

我们的任务是找到这个系统中温度与每个粒子定容热容 C 的关系. 由定义

$$C = \frac{\partial E_p}{\partial T} = k \cdot \frac{\partial E_p}{\partial \theta} \tag{16.8}$$

其中 E_p 是每个粒子内能，由下列方程给出

$$E_p = \frac{\lambda}{N} \int_0^\infty \epsilon \langle n(\epsilon) \rangle \mathrm{d}\epsilon = \frac{\lambda}{N} \cdot \theta^{5/2} \int_0^\infty \frac{x^{3/2} \mathrm{d}x}{\mathrm{e}^{x-\mu/\theta} - 1}. \tag{16.9}$$

定义函数 (通常称为 Bose-Einstein 积分，参见 [2])

$$F_{be}(z, \nu) = \frac{1}{\Gamma(\nu)} \int_0^\infty \frac{x^{\nu-1} \mathrm{d}x}{\mathrm{e}^{x+z} - 1}, \qquad z \geq 0, \quad \nu > 0; \tag{16.10}$$

正如会看到的， $F_{be}(0, \nu) = \zeta(\nu)$.

现在按下列步骤求解我们的问题:

1. 计算 μ 与温度的关系:

   对于 $\theta \leq \theta_c$: 令 $\mu = 0$,

   对于 $\theta \geq \theta_c$: μ 是方程 (16.3) 和 (16.6) 的解

$$N = \lambda \cdot \theta_c^{3/2} \cdot \Gamma\left(\frac{3}{2}\right) \cdot \zeta\left(\frac{3}{2}\right) = \lambda \cdot \theta^{3/2} \cdot \Gamma\left(\frac{3}{2}\right) \cdot F_{be}\left(-\frac{\mu}{\theta}, \frac{3}{2}\right), \tag{16.11}$$

   它导出方程

$$F_{be}\left(-\frac{\mu}{\theta}, \frac{3}{2}\right) = \left(\frac{\theta_c}{\theta}\right) \cdot \zeta\left(\frac{3}{2}\right). \tag{16.12}$$

   利用无量纲变量

$$y = \frac{\theta}{\theta_c}, \qquad \mu_d = \frac{\mu}{\theta_c},$$

   可把前面的方程转换成正则形式. 这样方程 (16.12) 变成

$$F_{be}\left(-\frac{\mu_d}{y}, \frac{3}{2}\right) = \frac{\zeta\left(\frac{3}{2}\right)}{y^{3/2}}. \tag{16.13}$$

2. 计算每个粒子的能量，由 (16.9)(也用到 (16.6)) 可得

$$E_p = \frac{\Gamma\left(\frac{5}{2}\right)}{\Gamma\left(\frac{3}{2}\right)} \cdot \frac{\theta^{5/2}}{\theta_c^{3/2}} \cdot \frac{1}{\zeta\left(\frac{3}{2}\right)} \begin{cases} \zeta\left(\frac{5}{2}\right) & \theta \leq \theta_c \\ F_{be}\left(-\frac{\mu}{\theta}, \frac{5}{2}\right) & \theta \geq \theta_c. \end{cases} \tag{16.14}$$

   使用无量纲量 $\epsilon_d = E_p/\theta_c$ 代替 E_p, 我们能够把 (16.14) 重写为

$$\epsilon_p = \frac{\Gamma\left(\frac{5}{2}\right)}{\Gamma\left(\frac{3}{2}\right)} \cdot \frac{y^{5/2}}{\zeta\left(\frac{3}{2}\right)} \begin{cases} \zeta\left(\frac{5}{2}\right) & y \leq 1 \\ F_{be}\left(z, \frac{5}{2}\right) & y \geq 1. \end{cases} \tag{16.15}$$

   其中 $z = -\mu_d/y$ 是方程 (16.13) 的解

$$y^{3/2} \cdot F_{be}\left(z, \frac{3}{2}\right) = \zeta\left(\frac{3}{2}\right). \tag{16.16}$$

3. 计算以 k 为单位的热容

$$c = \frac{C}{k} = \frac{\mathrm{d}\epsilon_d}{\mathrm{d}y}. \tag{16.17}$$

   我们注意到，当 $y >> 1$ 时，每个粒子能量的表达式 $E_p = 3\theta/2$ 成立，由此推得 $c = 3/2$.

16.2 MAPLE 解法

求解我们问题的关键是表示函数

$$F_{be}(z,\nu)=\frac{1}{\Gamma(\nu)}\int_0^\infty\frac{x^{\nu-1}\mathrm{d}x}{\mathrm{e}^{x+z}-1},\qquad z\geq 0,\nu\geq 0; \tag{16.18}$$

这个函数是黎曼 ζ 函数的一种推广并且不能在 MAPLE 中执行. 因为我们感兴趣的是在低温条件下, 当 $z\to 0$ 时系统的行为, 所以必须利用 F_{be} 对 z 的幂级数展开.

MAPLE 不能直接由方程 (16.18) 导出这个展开式. 因此, 我们必须利用对 $\nu>1$ 的如下展开式 ([2,3])

$$F_{be}(z,\nu)=z^{\nu-1}\Gamma(1-\nu)+\sum_{n=0}^{\infty}(-z)^n\frac{\zeta(\nu-n)}{n!} \tag{16.19}$$

它对所有的 $z\geq 0$ 都收敛. 我们定义函数 series/Fbe(z,nu,t)(参见算法 16.1), 它使用标准 MAPLE 函数 series(参见 MAPLE 的 Help) 执行 $F_{be}(z,n)$ 的级数展开对 t 的求值.

算法 16.1 函数 $F_{be}(z,\nu)$

```
`series/Fbe` := proc (z, nu::numeric, t) local n;
       # calculation of Bose-Einstein's  integral
       # for nu noninteger
       # 1/GAMMA(nu)*int(x^(nu - 1)/(e^(z + x) - 1), x = 0..inf)
  if type(nu, integer) or not (nu > 1) then
      ERROR('2-nd arg. expected to be noninteger and > 1');
  else
       # power expansion in z to order Order
      series((z)^(nu - 1)*GAMMA(1 - nu)
        + sum((-z)^n*Zeta(nu-n)/n!,n=0..Order -1)
        + O(t^Order),t);
  fi
end;
```

第二个问题是计算像 $\mu(\theta)$ 或 $E_p(\theta)$ 这样的函数, 它们具有两个不同的解析区域. 物理学家喜欢使用一个对象来表示一个物理量. 所以为了定义上述量, 最好的方法是用 *if − then − else* 语句来定义函数, 例如

$$If(<relation>,<vthen>,<velse>),$$

它根据 $<relation>$ 的布尔值返回 $<vthen>$ 或 $<velse>$ 表达式 (新版 MAPLE 将支持定义分段函数). 由于我们需要计算这个函数的导数, 因此可通过定义函数 diff/If(参见 MAPLE 的 Help) 来使用标准函数 diff. 在算法 16.2 中定义了这些函数.

求解问题的第一步是对 z 求解方程 (16.16). 在这个方程中 $F_{be}(z,3/2)$ 以 z 的级数形式来表示, 它也包含 $z^{1/2}$ 项. 但 MAPLE 仅能解纯幂级数的方程. 为此我们作变换 $z\to w^2$, 然后对 w 求解变换后的方程

$$y^{3/2}\cdot F_{be}\left(w^2,\frac{3}{2}\right)=\zeta\left(\frac{3}{2}\right). \tag{16.20}$$

MAPLE 能够解出这个方程并以 $x = 1 - y^{-3/2}$ 的级数形式给出 z. 这样我们能够利用函数 If 把化学势 μ_d 表示成

$$If(y \leq 1, 0, -y \star z).$$

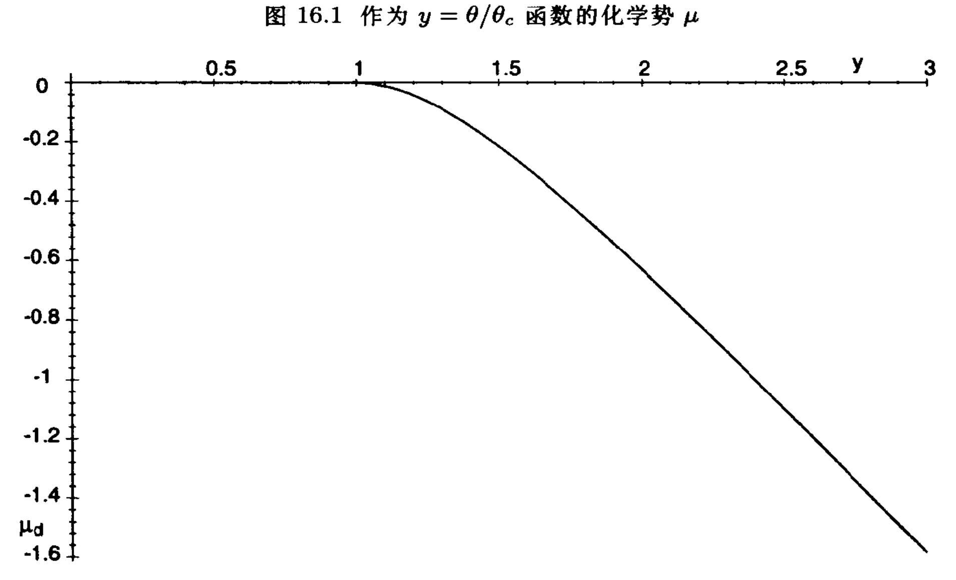
算法 16.2 函数 If

```
# syntax of calling:  If(bool,then,else)

`diff/If` := proc(bool, vthen, velse, x) ;
  If(bool, diff(vthen, x), diff(velse, x))
end;

If:= proc(bool, vthen, velse)
  if type(bool, 'relation'(numeric)) then
    if bool then vthen else velse fi
  else 'If(args)'
  fi
end;
```

图 16.1 作为 $y = \theta/\theta_c$ 函数的化学势 μ



这个函数的图像如图 16.1. 所需的 MAPLE 语句如下:

```
>            # Basic equation for the chem.  potential mud
>            # for y >= 1; z = -mud/y
> Eq := series(Fbe(z, 3/2), z, 4) = y^(-3/2)*Zeta(3/2);
```

$$\boldsymbol{Eq} := (-2\sqrt{z}\sqrt{\pi} + \zeta(\frac{3}{2}) - z\,\zeta(\frac{1}{2}) + \frac{1}{2}z^2\zeta(\frac{-1}{2}) - \frac{1}{6}z^3\zeta(\frac{-3}{2}) + \mathrm{O}(z^4)) = \frac{\zeta(\frac{3}{2})}{y^{3/2}}$$

```
>            # we choose the substitution
>            # y^(-3/2) -> 1 - x and z-> w^2
>            # and invert the series
>  solw := solve(series(Fbe(w^2, 3/2), w, 7)
>           = (1 - x)*Zeta(3/2), w):
>            # then
>  z := series(solw^2, x, 4); # because z = w^2 (z = -mu/theta);
```

$$z := \frac{1}{4}\frac{\zeta(\frac{3}{2})^2}{\pi}x^2 - \frac{1}{8}\frac{\zeta(\frac{3}{2})^3\zeta(\frac{1}{2})}{\pi^2}x^3 + \mathrm{O}(x^4)$$

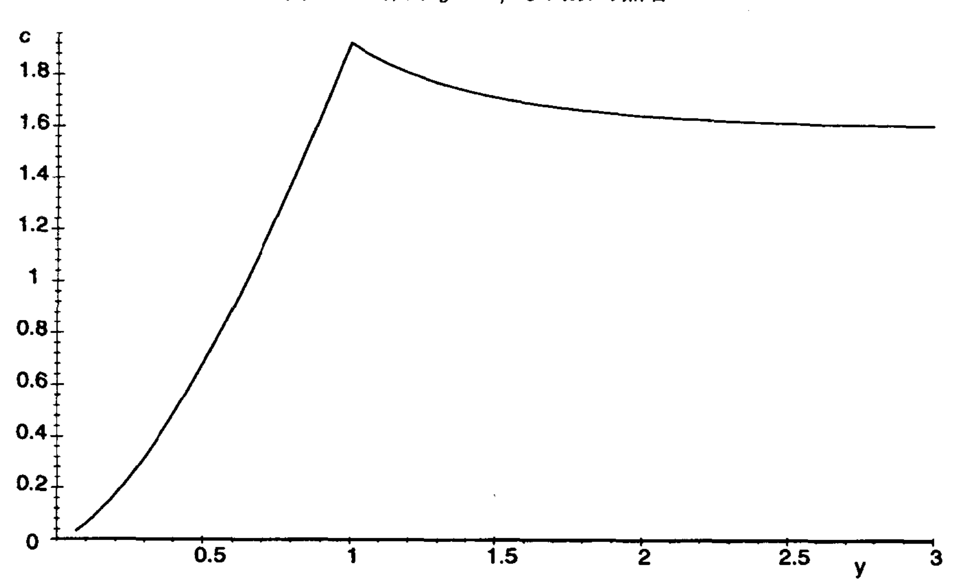
```
>  z:= convert(z, polynom):  # for next evaluation
>            # graf of mud=-z*y on y
>  plot(If(y <= 1, 0, -y*subs(x = 1 - y^(-3/2), z)), y = 0..3);
```

无量纲的内能由方程 (16.15) 给出，我们能够将其表示成

$$\epsilon_d = \frac{3}{2}\frac{y^{5/2}}{\zeta\left(\frac{3}{2}\right)} \cdot If\left(y \le 1, \zeta\left(\frac{5}{2}\right), F_{be}\left(z, \frac{5}{2}\right)\right) \tag{16.21}$$

这里 $F_{be}(z, 5/2)$ 用 z 表示，而 z 在前一步已经作为 $x = 1 - y^{-3/2}$ 的级数被表示出来．热容 c 由方程 (16.17) 给出，

图 16.2 作为 $y = \theta/\theta_c$ 函数的热容 c



这个函数 $c(y)$ 的图像如图 16.2 所示．我们应该想到，$c(y)$ 的导数在 $y = 1$ 点有间断；这表明 $(\theta = \theta_c)$ 这一点是二阶的相变点． MAPLE 程序的其余部分如下：

```
>            # calculation of dimensionless internal energy
>            # per particle Ed, expansion of Fbe(z,5/2) in y
>  Fy := subs(x = 1 - y^(-3/2),
```

```
>            convert(series(Fbe(z, 5/2), x, 4), polynom));
```

$$Fy := \zeta(\frac{5}{2}) - \frac{1}{4}\,\frac{\zeta(\frac{3}{2})^3\,(1-\frac{1}{y^{3/2}})^2}{\pi} + \left(\frac{1}{6}\,\frac{\zeta(\frac{3}{2})^3}{\pi} + \frac{1}{8}\,\frac{\zeta(\frac{3}{2})^4\,\zeta(\frac{1}{2})}{\pi^2}\right)\,(1-\frac{1}{y^{3/2}})^3$$

```
>          # using the function If we can express energy for all y
>   Ed := 3/2/Zeta(3/2)*y^(5/2)*If(y <= 1, Zeta(5/2), Fy):
>          #heat capacity is (in units of k)
>   C := diff(Ed, y):
>          #and than we make graph of heat capacity
>   plot(C, y = 0..3);
```

参考文献

[1] C. KITTEL, *Elementary Statistical Physics*, John Wiley & Sons, Inc., 1958.

[2] F. LONDON, *Superfluids II.*, John Wiley & Sons, Inc., 1954.

[3] J. E. ROBINSON, *Note on the Bose-Einstein integral functions*, Physical Review, 83, 1951, pp. 678-679.

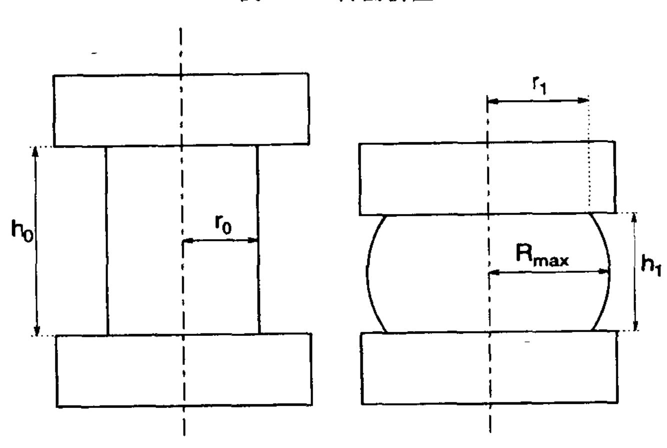
第十七章　金属自由挤压加工

S. Bartoň

17.1　引言

挤压是金属成形中广泛应用的加工手段. 如果挤压是采用两块平面压板 (挤压的一个特例), 金属的侧面将变形 (参见图 17.1). 这种连续成形操作叫做模具成形, 它必须提前预测变形的程度, 以便给在特定模具中变形的物体提供足够的空间. 本章只讨论横截面不变的金属棒材, 因为横截面变化的情况在实际中不会涉及到.

图 17.1　自由挤压



我们将在如下假定下讨论被挤压的物体:

1. 物体体积 V 保持不变
$$V = S_0 H = \text{const.}, \tag{17.1}$$
此处 S_0 = 初始底部面积, H = 棒材的初始高度.
2. 初始的棒材用柱坐标描述. 棒材的长度方向平行于 z 轴, 坐标系的原点在棒材底部的质心上 (参见图 17.2)
3. 材料只有径向和轴向的移动, 没有切向的移动.
4. 如果物体开始是关于 z 轴对称的, 则形变后的物体也保持同样的对称性.
5. 描述底部周长的径向距离函数 $\rho(\phi,h)$ 满足微分方程
$$\rho_h(\phi,h) \equiv \frac{\partial\rho(\phi,h)}{\partial h} = C(h)k\frac{1}{\rho(\phi,h)}, \tag{17.2}$$
此处 h 是被挤压物体的当前高度, $C(h)$ 是待定的比例因子, k 描述压板和棒材之间的磨擦大小:
(a) $k=1$: 无磨擦. 侧面平行于 z 轴. 底部达到最大限度的扩展. 这时有
$$S = S_0\sqrt{H/h},$$

S 表示当前的底部面积， H 是初始的高度.

(b) $k=0$: 磨擦无限大. 底部保持其初始形状:

$$S = S_0 = \text{const}.$$

侧面的变形最大.

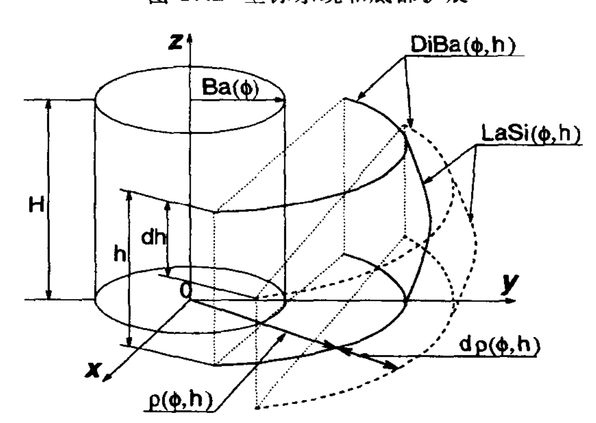
(c) $0 < k < 1$. 两个底部都有扩展，侧面也有变形.

$$S_0 < S < S_0\sqrt{H/h}.$$

K 的典型值是 $0.3 \le k \le 0.7$.

6. 变形物体的任何轴向截面都产生一个对称曲线；我们把它近似成抛物线 (参见 [4]).

图 17.2 坐标系统和底部扩展



17.2 底部的扩展

在用 MAPLE 求解方程 (17.2) 之前，我们用 (17.1) 来确定未知函数 $C(h)$.

```
> de := diff(rho(h), h) = C(h)*k/rho(h):
> dsolve(de, rho(h));
```

$$\rho(h)^2 = 2\,k\int \mathrm{C}(h)\,dh + _C1$$

```
> Sol_de := solve(", rho(h));
```

$$Sol_de := \sqrt{2\,k\int \mathrm{C}(h)\,dh + _C1},\ -\sqrt{2\,k\int \mathrm{C}(h)\,dh + _C1}$$

```
> DiBa := Sol_de[1]:
```

为了简化计算，引入新变量 ICh 代替 $\int C(h)\mathrm{d}h$. 积分常数 $_C1$ 根据未压缩的棒材决定:

$$\mathrm{DiBa}(\phi, H) = \mathrm{Ba}(\phi),$$

此处 Ba(ϕ) 是用极坐标表示的底部周长 (见图 17.2). 作替换

$$\mathrm{NH} := \int_0^H C(h)\mathrm{d}h$$

```
> DiBa := Sol_de[1]:
> DiBa := subs(int(C(h), h) = ICh, DiBa):
> E1 := subs(ICh = NH, DiBa) = Ba(phi):
> _C1:=solve(E1,_C1);
```

$$_C1 := -2\,k\,NH + \mathrm{Ba}(\phi)^2$$

如果平板与棒材之间没有磨擦, 方程 (17.1) 可写成如下形式

$$h\int_{-\pi}^{\pi}\frac{\mathrm{DiBa}(\phi,h)^2}{2}\mathrm{d}\phi = H\int_{-\pi}^{\pi}\frac{\mathrm{Ba}(\phi)^2}{2}\mathrm{d}\phi,$$

因为侧面没有变形, 且横截面是常数. ICh 现在可用这个方程来确定:

```
> Eq := h*Int(subs(k = 1, DiBa)^2/2, phi = -Pi..Pi)
>     = H*Int(Ba(phi)^2/2, phi = -Pi..Pi):
> EI1 := ICh = value(solve(Eq, ICh));
```

$$EI1 := ICh = \frac{1}{4}\,\frac{4\,h\,NH\,\pi - h\displaystyle\int_{-\pi}^{\pi}\mathrm{Ba}(\phi)^2\,d\phi + H\int_{-\pi}^{\pi}\mathrm{Ba}(\phi)^2\,d\phi}{h\,\pi}$$

由于 S_0 是棒材的初始横截面积, 为简化问题, 现在我们用众所周知的极坐标公式计算封闭曲线包围的面积.

$$S_0 = \int_{-\pi}^{\pi}\frac{\mathrm{Ba}(\phi)^2}{2}\mathrm{d}\phi$$

在第二步, 我们计算最后结果的导数, 得到 $C(h)$ 的表达式:

```
> EI2 := subs(int(Ba(phi)^2, phi = -Pi..Pi) = 2*So, EI1):
> EI3 := subs(ICh = int(C(h), h), EI2);
```

$$EI3 := \int \mathrm{C}(h)\,dh = \frac{1}{4}\,\frac{4\,h\,NH\,\pi - 2\,h\,So + 2\,H\,So}{h\,\pi}$$

```
> EI4 := normal(diff(EI3, h));
```

$$EI4 := \mathrm{C}(h) = -\frac{1}{2}\,\frac{H\,So}{h^2\,\pi}$$

现在问题已简化为计算 NH $= C(h)$, 而 DiBa(ϕ, h) 就确定了.

```
> NH := subs(h = H, rhs(EI4));
```

$$NH := -\frac{1}{2}\,\frac{So}{H\,\pi}$$

将它代入微分方程可以检验结果的正确性.

```
> DiBa:=simplify(subs(ICh = rhs(EI3), DiBa));
```

$$DiBa := \sqrt{\frac{-k\,So\,h + k\,So\,H + \mathrm{Ba}(\phi)^2\,h\,\pi}{h\,\pi}}$$

```
> de_test := simplify(subs(rho(h) = DiBa, C(h) = rhs(EI4), de), symbolic):
> evalb(");
```

$$true$$

把这个解应用到无磨擦的金属园柱体的特例. 在这个特例中, 由方程 (17.1) 导出

$$\mathrm{DiBa}(h) = \sqrt{n}r,$$

此处 $r=$ 柱体的初始半径, 而 $n = H/h =$ 压缩比.

```
> DiBa:=subsop(1 = collect(expand(op(1, DiBa)), [So, Pi, k]),DiBa);
```

$$DiBa := \sqrt{\frac{(-1+\frac{H}{h})\,k\,So}{\pi} + \mathrm{Ba}(\phi)^2}$$

```
> Cylinder_test := evalb(r*sqrt(n) = simplify(subs(So = Pi*r^2,
>                  Ba(phi) = r, h = H/n, k = 1, DiBa),symbolic));
```

$$Cylinder_test := true$$

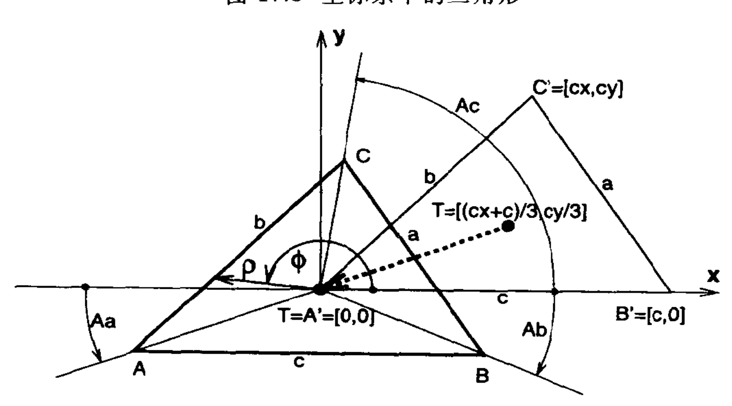
```
> save(DiBa, 'DiBa.sav');
```

17.3 由一个或几个函数描述的底部

在本节中, 我们考虑函数族 $\mathrm{Ba}(\phi)$ 而不是象以前的 $Ba(\phi) =$ 常数. 在这种一般性的情形下, 底部的周长是椭园形或三角形; 注意三角形的解很容易应用到任何分割后的底部.

通常在技术上三角形由它的边长 a, b, c 来描述. 为达到目的, 我们必须用极坐标描述三角形的顶点, 而坐标的原点必须在三角形的质心. 可以按如下步骤来做: 旋转三角形使得 c 边平行于 x 轴; 然后移动三角形使得它的质心在坐标系的原点. 最后计算顶角以确定两边的夹角 ϕ. 参考图 17.3 中关于平移的图示.

图 17.3 坐标系中的三角形




```
> restart;
```

```
> with(linalg): with(plots):
> e1 := cx^2 + cy^2 = b^2:
> e2 := (cx - b)^2 + cy^2 = a^2:
> allvalues(solve({e1, e2}, {cx, cy}));
```

$$\{cx=-\frac{1}{2}\frac{-2b^2+a^2}{b},\ cy=\frac{1}{2}\frac{\sqrt{4b^2-a^2}\,a}{b}\},\ \{cx=-\frac{1}{2}\frac{-2b^2+a^2}{b},\ cy=-\frac{1}{2}\frac{\sqrt{4b^2-a^2}\,a}{b}\}$$

```
> assign("[1]);
> A := [0, 0]:  B := [c, 0]:  C := [cx, cy]:
> T := (A + B+ C)/3:
> A := normal(A - T): B := normal(B - T): C := normal(C - T):
> Aa := abs(arctan(A[2]/A[1])):
> Ab := abs(arctan(B[2]/B[1])):
> Ac := Pi/2 - arctan(C[1]/C[2]):
```

为了导出三角形边的方程，我们利用了极坐标中由两个顶点 $P\equiv[P_x,P_y]$ 和 $Q\equiv[Q_x,Q_y]$ 确定的直线的一般方程. 描述各边的线段由角 Aa,Ab 和 Ac 来进一步表示.

```
> rho := (Q[y]*P[x] - Q[x]*P[y])/
>         ((P[x] - Q[x])*sin(phi) -(P[y] - Q[y])*cos(phi));
```

$$\rho:=\frac{Q_y\,P_x-Q_x\,P_y}{(P_x-Q_x)\sin(\phi)-(P_y-Q_y)\cos(\phi)}$$

```
> r1 := normal(subs(P[x] = A[1], Q[x] = B[1], P[y] = A[2],
>                   Q[y] = B[2], rho));
> r2 := normal(subs(P[x] =B[1], Q[x] = C[1], P[y] = B[2],
>                   Q[y] = C[2], rho));
> r3 := normal(subs(P[x] = C[1], Q[x] = A[1], P[y] = C[2],
>                   Q[y] = A[2], rho));
```

$$r1:=-\frac{1}{6}\frac{\sqrt{4b^2-a^2}\,a}{b\sin(\phi)}$$

$$r2:=\frac{1}{3}\frac{\sqrt{4b^2-a^2}\,a\,c}{-2\sin(\phi)\,b^2+\sin(\phi)\,a^2+2\sin(\phi)\,c\,b+\sqrt{4b^2-a^2}\,a\cos(\phi)}$$

$$r3:=-\frac{1}{3}\frac{\sqrt{4b^2-a^2}\,a\,c}{-2\sin(\phi)\,b^2+\sin(\phi)\,a^2+\sqrt{4b^2-a^2}\,a\cos(\phi)}$$

函数 $\mathrm{Ba}(\phi)$ 由三直线 r_1,r_2,r_3 组成. 为了计算 S_0 我们利用了 Heron 法则.

```
> o := (a + b + c)/2:
> Sb := sqrt(o*(o - a)*(o - b)*(o - c)):
> save(r1, r2, r3, Sb, Aa, Ab, Ac, 'Tri.sav');
```

重要的变量存在文件 `Tri.sav` 中待以后用.

接下来还要代入 a,b,c 的值. 让我们观察一下由 $a=8,b=7,c=6$ 描述的三角形棒材的伸展. 对磨擦系数的三个值 $k=1/3,2/3,1$ 以及一系列压缩反比的值 $h/H:=n=1,0.9,\ldots,0.1$, 我们将画出三角形的底部扩展. 为了比较这个形变我们也画出了横截面为椭园的棒材的底部扩展, 椭园的半长轴为 $a=3.5$, 半短轴为 $b=1.85$, 以使两个棒材的面积相等.

为了用极坐标描述椭园，我们使用

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1 \quad \Longrightarrow \quad \rho=\frac{ab}{\sqrt{a^2\sin(\phi)^2+b^2\cos(\phi)^2}}$$

```
> Sus := a = 8, b = 7, c = 6:  Suf := h = N*H, So = Sb:
> Aa := evalf(subs(Sus, Aa)):
> Ab := evalf(subs(Sus, Ab)):
> Ac := evalf(subs(Sus, Ac)):
> R1 := evalf(subs(Suf, Ba(phi) = r1, Sus, DiBa)):
> R2 := evalf(subs(Suf, Ba(phi) = r2, Sus, DiBa)):
> R3 := evalf(subs(Suf, Ba(phi) = r3, Sus, DiBa)):
> RE := subs(h = N*H, Ba(phi) = a*b/sqrt(a^2*sin(phi)^2
>       + b^2*cos(phi)^2), So = Pi*a*b, a = 3.5, b = 1.85, DiBa):
> SqN := [seq(1 - i*0.1, i = 0..8)]:
> with(plots):
```

下面的 MAPLE 程序画出图 17.4 中的图形.

```
> for k from 1/3 by 1/3 to 1 do:
>   P1 := polarplot({seq(R1, N = SqN)}, phi= - Pi + Aa..-Ab):
>   P2 := polarplot({seq(R2, N = SqN)}, phi = -Ab..Ac):
>   P3 := polarplot({seq(R3, N = SqN)}, phi = Ac..Pi + Aa):
>   print(display({P1, P2 ,P3}, scaling = constrained,
>           view = [-6.5..6.5, -6..7]));
>   print(polarplot({seq(RE, N = SqN)}, phi = -Pi..Pi,
>               scaling = constrained, view = [-6.5..6.5, -6..7]));
> od:
> k := 'k':
```

17.4 侧面的形变

由于材料是径向移动，就象我们说过的那样，可以简化对侧面形变的计算. 只有轴向截面的边界线需要计算. 所用的函数如图 17.5 所描述. 在柱坐标中，侧面的形状假设为抛物线，

$$\mathrm{LaSi}(\phi,z)=Q_2(\phi)z^2+Q_1(\phi)z+Q_0(\phi), \tag{17.3}$$

这里要确定函数 $Q_2(\phi),Q_1(\phi)$ 和 $Q_0(\phi)$. 从图 17.5 可看到，

$$\begin{aligned}\mathrm{LaSi}(\phi,0) &= Q_0(\phi)\\ \mathrm{LaSi}(\phi,h) &= Q_0(\phi).\end{aligned}$$

进一步，将方程 (17.1) 中的常数表示为

$$\iiint \mathrm{d}V = S_0H.$$

由于函数 $\mathrm{Ba}(\phi)$ 尚未指定，我们利用 Fubini 定律来交换积分的顺序：

$$\iiint \mathrm{d}V=\int_{-\pi}^{\pi}\left(\int_0^h\frac{\mathrm{LaSi}(\phi,z)^2}{2}\mathrm{d}z\right)\mathrm{d}\phi$$

图 17.4 三角形和椭圆底部的形变

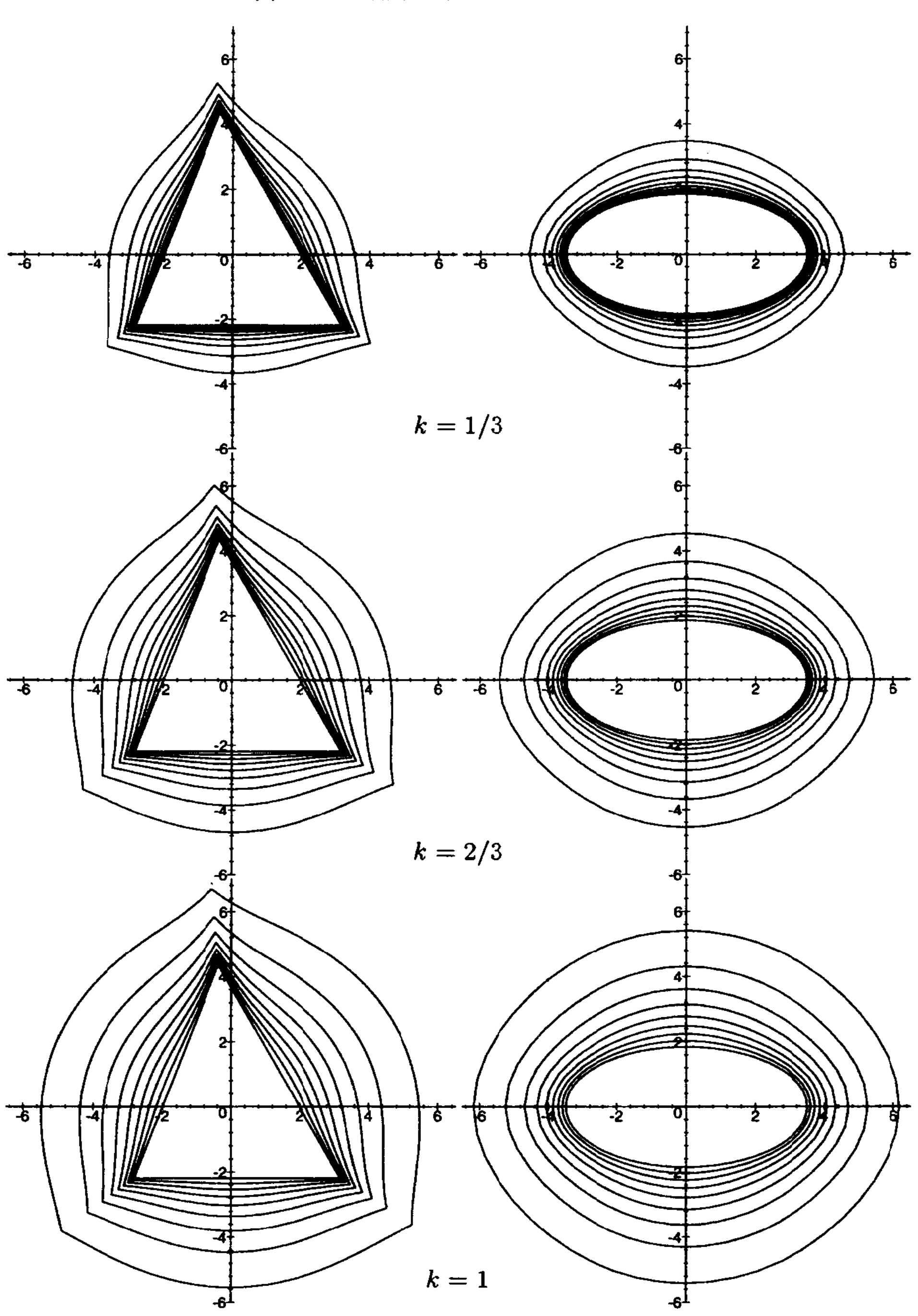
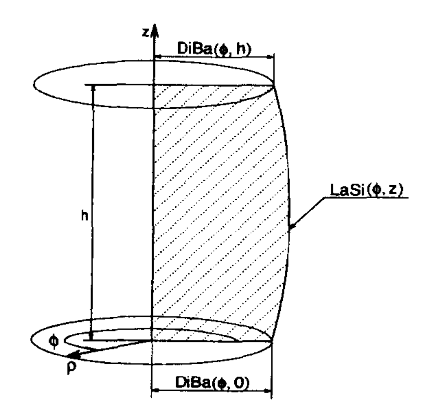


图 17.5 侧面的形状



为了方便阅读，以下推导中的 DiBa 仅在最后的表达式中以下式代入：

$$\mathrm{DiBa} \equiv \sqrt{\frac{(-1+\frac{H}{h})kS_0}{\pi} + \mathrm{Ba}(\phi)^2}$$

```
> restart;
> LaSi := Q2*z^2 + Q1*z + Q0:
> E1 := subs(z = 0, LaSi) = DiBa;
```

$$E1 := Q0 = DiBa$$

```
> assign(");
> E2 := subs(z = h, LaSi) = DiBa:
> Q1 := solve(E2, Q1);
```

$$Q1 := -Q2\,h$$

```
> E3 := Int(Int(LaSi^2/2, z = 0..h), phi = -Pi..Pi) = H*So;
```

$$E3 := \int_{-\pi}^{\pi}\int_{0}^{h}\frac{1}{2}\,(Q2\,z^2 - Q2\,h\,z + DiBa)^2\,dz\,d\phi = H\,So$$

为了计算双重积分，我们分几步进行．先对 z 积分

```
> I1 := int(LaSi^2/2, z = 0..h);
```

$$I1 := \frac{1}{60}\,Q2^2\,h^5 - \frac{1}{6}\,DiBa\,Q2\,h^3 + \frac{1}{2}\,DiBa^2\,h$$

然后将和式分为三个积分：

```
> I2 := subsop(seq(i = Int(op(i, I1), phi = -Pi..Pi),
>             i = 1..nops(I1)), I1);
```

$$I2 := \int_{-\pi}^{\pi}\frac{1}{60}\,Q2^2\,h^5\,d\phi + \int_{-\pi}^{\pi}-\frac{1}{6}\,DiBa\,Q2\,h^3\,d\phi + \int_{-\pi}^{\pi}\frac{1}{2}\,DiBa^2\,h\,d\phi$$

```
> I21 := subsop(2 = h/2*Int(DiBa^2, phi = -Pi..Pi),
>          3 = -Q2*h^3/6*Int(DiBa, phi = -Pi..Pi), I2);
```

$$I21 := \int_{-\pi}^{\pi} \frac{1}{60}\, Q2^2\, h^5\, d\phi + \frac{1}{2}\, h \int_{-\pi}^{\pi} DiBa^2\, d\phi - \frac{1}{6}\, Q2\, h^3 \int_{-\pi}^{\pi} DiBa\, d\phi$$

再将底部形变函数代入积分并化简:

```
> read('DiBa.sav'):
> I22 := subsop(1 = value(op(1, I21)),
>                2 = -So*(k*h - k*H - h), I21);
```

$$I22 := \frac{1}{30}\, \pi\, Q2^2\, h^5 - So\,(k\,h - k\,H - h) - \frac{1}{6}\, Q2\, h^3 \int_{-\pi}^{\pi} \sqrt{\frac{(-1 + \frac{H}{h})\, k\, So}{\pi} + \mathrm{Ba}(\phi)^2}\, d\phi$$

```
> E3 := subs(op(4, op(3, I22)) = IDiBa, I22) = H*So;
```

$$E3 := \frac{1}{30}\, \pi\, Q2^2\, h^5 - So\,(k\,h - k\,H - h) - \frac{1}{6}\, Q2\, h^3\, IDiBa = H\, So$$

这样得到一个关于 $Q2$ 的二次方程, 它有两个解. 为了确定在物理上有意义的解, 我们把两个解 $Q2_1, Q2_2$ 都代入 LaSi, 最后通过观察图 17.6 来确定.

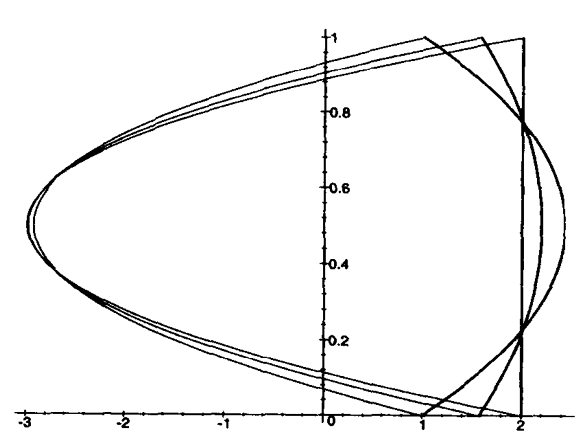
```
> Sol := solve(E3, Q2):
> T1 := Sol[1]:  T2 := Sol[2]:
> LaSi_1:=[subs(Q2 = T1, LaSi), subs(Q2 = T2, LaSi)]:
> Cylinder_Subs := IDiBa = int(DiBa, phi = -Pi..Pi),
>     So = Pi*r^2, Ba(phi) = r, r=1 , H = 4, k = kappa, h = 1:
> LaSi_2 := simplify(subs(Cylinder_Subs, LaSi_1)):
> Kappa := [seq(1 - i/2, i = 0..2)]:
> with(plots):
> P1 := plot({seq([LaSi_2[1], z, z = 0..1], kappa = Kappa)},
>              thickness = 3):
> kappa := 'kappa':
> P2 := plot({seq([LaSi_2[2], z, z = 0..1], kappa = Kappa)},
>              thickness = 1):
> kappa := 'kappa':
> display({P1, P2});
```

把正确的根 $Q2_1$ 代入 LaSi, 得到非常复杂的表达式. 在用 MAPLE 简化了部分表达式之后, 求出了最终的结果, 它可写为如下形式:

```
> LaSi := -z*(-z + h)/(2*h^2*Pi)*(5*Int(DiBa, phi = A..B)
>          -sqrt(5)*sqrt(5*Int(DiBa, phi = A..B)^2
>          - 24*Pi*So*(k - 1)*(-1 + H/h))) + DiBa:
> subs (DiBa='DiBa', LaSi);
```

$$\mathrm{LaSi} := -\frac{1}{2}\, \frac{z\,(-z+h)\,(5\int_A^B DiBa\, d\phi - \sqrt{5}\sqrt{5\,(\int_A^B DiBa\, d\phi)^2 - 24\,\pi\, So\,(-1+\frac{H}{h})\,(k-1)})}{\pi\, h^2} + DiBa$$

图 17.6 对 $Q2_1$ 和 $Q2_2$ 的柱面测试



其中 A, B 是描述线段的角.

```
> save(LaSi, 'LaSi.sav'):
```

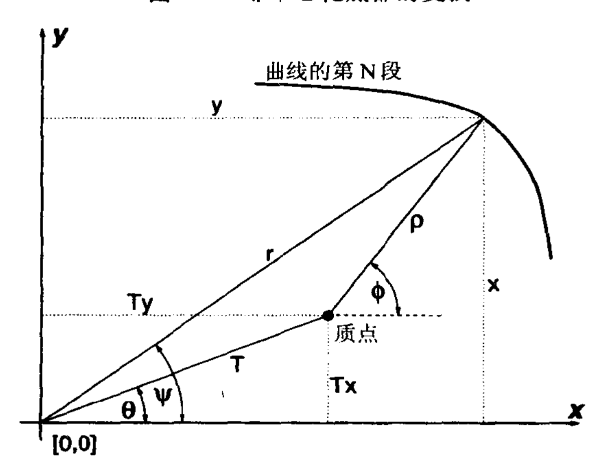
17.5 非中心化的底部

我们定义非中心化的底部是指质心不在坐标系原点的底部. 对于这种底部需要移动坐标系以使质心与坐标系的原点重合. 在直角坐标中这是一个极简单的问题, 然而在极坐标中就不简单了(见图 17.7). 非中心化的底部周长由 $r(\psi)$ 来描述. 但我们需要的是函数 $\rho(\phi)$. 我们必须用新的变量将积分表示为

$$\int_A^B \sqrt{\frac{(-1+\frac{H}{h})kS_0}{\pi} + Ba(\phi)^2 \mathrm{d}\phi}. \tag{17.4}$$

所以必须找到函数 r_x, ψ_x 和 $\mathrm{d}\psi_x$, 它们分别满足方程 $\rho(\phi) = r_x(\psi), \phi = \psi_x(\psi)$ 和 $\mathrm{d}\phi = \mathrm{d}\psi_x(\psi)$.

图 17.7 非中心化底部的变换



```
> restart;
> Tx := T*cos(theta): Ty := T*sin(theta):
> r1x := rho(phi)*cos(phi) + Tx = r(psi)*cos(psi):
> r1y := rho(phi)*sin(phi) + Ty = r(psi)*sin(psi):
> r2x := map(t -> t - Tx, r1x): r2y := map(t -> t - Ty, r1y):
> R1 := combine(map(t -> t^2, r2x) + map(t -> t^2, r2y)):
> sol := rho(phi) = combine(solve(R1, rho(phi))[1]);
```

$$sol := \rho(\phi) = \sqrt{\mathrm{r}(\psi)^2 - 2\,\mathrm{r}(\psi)\,T\cos(\psi-\theta) + T^2}$$

在给定一个角和相邻的两边以后，可得到三角形第三边的表达式 (余弦定理). 现在可以将 ϕ 和 $\mathrm{d}\phi$ 表为 ψ 的函数. 为了方便，定义 $\cos(\phi) = \mathrm{Cf}(\psi)$ 和 $\sin(\phi) = \mathrm{Sf}(\psi)$, 因为需要用这些表达式来作图.

```
> R2 := phi = solve(lhs(r2y)/lhs(r2x) = rhs(r2y)/rhs(r2x), phi):
> dR2 := D(phi) = normal(diff(rhs(R2),psi)):
> df := factor(combine(rhs(dR2)));
```

$$df := \frac{T\,(\frac{\partial}{\partial\psi}\,\mathrm{r}(\psi))\sin(\psi-\theta) - \mathrm{r}(\psi)^2 + \mathrm{r}(\psi)\,T\cos(\psi-\theta)}{-\mathrm{r}(\psi)^2 + 2\,\mathrm{r}(\psi)\,T\cos(\psi-\theta) - T^2}$$

```
> Cf := solve(r2x, cos(phi)): Sf := solve(r2y, sin(phi)):
> Cf := subs(sol, Cf); Sf := subs(sol, Sf);
```

$$Cf := \frac{\mathrm{r}(\psi)\cos(\psi) - T\cos(\theta)}{\sqrt{\mathrm{r}(\psi)^2 - 2\,\mathrm{r}(\psi)\,T\cos(\psi-\theta) + T^2}}$$

$$Sf := -\frac{-\mathrm{r}(\psi)\sin(\psi) + T\sin(\theta)}{\sqrt{\mathrm{r}(\psi)^2 - 2\,\mathrm{r}(\psi)\,T\cos(\psi-\theta) + T^2}}$$

再将 Ba 和 $\mathrm{d}\phi$ 的表达式代入积分 (17.4). 如果底部的周长由具有不同函数描述的几个线段组成，那么必须将积分分成几段来计算.

为了优化各段计算，我们使用如下的 MAPLE 语句 (`FO` 优化了积分外部的代码， `FI` 优化了被积函数):

```
> read(`LaSi.sav`): read(`DiBa.sav`):
> FRzS := subs(Int(DiBa, phi = A .. B) =
>     Int(DiBa*df, psi = A ..B), Ba(phi) = Ba(psi), LaSi):
> readlib(optimize):
> FO := optimize([W1 = FRzS, W2 = Cf, W3 = Sf]):
> FI := optimize(op(1,rhs(op(1, select(has, [FO], Int))))):
```

然后消去各段共同的某些代码，得到最终的算法 17.1. 在调用程序 17.1 之前一件重要的事情是指定变量 EIco. 这个变量决定了数值精度和在积分计算中使用的数值方法. 通常 EIco 有如下的值:

```
> EIco := NDigits:
> EIco := NDigits, _NumMethod;
```

其中 `NDigits` 决定了所需的数值精度， `_NumMethod` 定义的积分方法可作如下替代.

$$\text{_NumMethod} = \begin{array}{ll} _CCquad & \text{(Clenshaw-Curtis quadrature)} \\ _Dexp & \text{(double exponential routine)} \\ _NCrule & \text{(Newton-Cotes rule)} \end{array}$$

最佳的选择取决于应用．要获得更详细的信息可输入

```
>  ?evalf[int]
```

算法 17.1 要求的输入是：

1. $R :: list$ 是描述底部周边线段的函数列表．线段按反时针排序．

$$R := [u \to f_1(u), u \to f_2(u), \cdots, u \to f_n(u)]$$

2. $Bo :: list$ 是每条线段的初始点的位置角加上最后一条线段的最后一个点的位置角的列表．

$$Bo := [\alpha_1, \alpha_2, \cdots, \alpha_n, \alpha_{n+1}] \qquad \alpha_{n+1} \equiv \alpha_1 + 2\pi$$

3. Hi, hf, kf 分别是未压缩的初始高度，压缩后的最终高度和磨擦力大小．

如果 R,Bo 和 So 已经定义，并且底部曲线已经中心化，就可调用算法 17.1:

```
>  T := 0:  theta := 0:
>  EIco := 12, _Dexp:
>  n := nops(R):
>  Body(R, Bo, H, h, k):
```

这个过程返回一个函数列表，它用柱坐标描述每个侧边的变形，假定底部已经中心化，

$$[(\phi, z) \to f_i(\phi, z)] \quad i = 1, \cdots, n.$$

如果没有中心化，得到的是一个带参数的函数，

$$[(\psi, z) \to f_i(\phi(\psi), z)] \quad i = 1, \cdots, n.$$

算法 17.1 非中心化 n 段曲线的优化算法

```
Body := proc (R::list, Bo::list, Hi, hf, kf)
  local i, q1, q2, q3, q4, q5, q6, w1, w2, w3, w4, w5,
        w6, w7, w11, w8, w9, w10, w12, w13, w14;
   global Rho, Cf, Sf, X, Y;
   q3 := -u + theta;  q5 := T^2;  w2 := 0;
   for i to n do
     q1[i] := R(u)[i];  q2[i] := q1[i]^2;
     q4[i] := q1[i]*T*cos(q3);
     q6[i] := ((1/hf*Hi - 1)/Pi*kf*So + q2[i] - 2*q4[i]
              + q5)^(1/2);
     w2 := w2 + evalf(Int(q6[i]*(T*diff(R(u)[i], u)*sin(q3)
              + q2[i] - q4[i])/(q2[i] - 2*q4[i] + q5),
                u = Bo[i].. Bo[i+1], EIco));
   od;
   w1 := hf^2;  w3 := 5^(1/2);   w4 := w2^2;
   w5 := -1 + 1/hf*Hi;
```

```
    w6 := (5*w4 - 24*Pi*So*w5*(-1 + kf))^(1/2);
    w7 := 1/Pi;  w11 := T^2;
    for i to n do
       w8[i] := R(u)[i];  w9[i] := w8[i]^2;
       w10[i] := w8[i]*T*cos(q3);
       w12[i] := (w5*w7*kf*So + w9[i] - 2*w10[i] + w11)^(1/2);
       w13[i] := (w9[i] - 2*w10[i] + w11)^(1/2);
       w14[i] := 1/w13[i];
    od;
    Rho := [seq(unapply(1/2*z*(-z + hf)/w1*(-5*w2 + w3*w6)*w7
                        + w12[i], u, z), i = 1 .. n)];
    Cf := [seq(unapply((w8[i]*cos(u) - T*cos(theta))*w14[i], u),
                        i = 1 .. n)];
    Sf := [seq(unapply((w8[i]*sin(u) - T*sin(theta))*w14[i], u),
                        i = 1 .. n)];
    X := [seq(unapply(Rho(u, z)[i]*Cf(u)[i], u, z), i = 1 .. n)];
    Y := [seq(unapply(Rho(u, z)[i]*Sf(u)[i], u, z), i = 1 .. n)];
    print(`To display the compressed body use the Maple command:`);
    if T = 0 then
      print(`cylinderplot(r(f,z), f=a..b, z=0..h)`);
    else
      print(`plot3d([x(f,z), y(f,z), z], f=a..b, z=0..h)`);
    fi;
end;
```

在第二种情况下，我们也使用函数列表 $[X_i], [Y_i], i = 1, \cdots, n$, 它是带 ψ 的参数函数，用来描述在直角坐标下的形变物体.

如果底部是非中心化的，我们必须先计算质心的极坐标 (T, θ), 这可用程序 17.2 来做，输入的参数 $\rho :: list$ 和 $bords :: list$ 与算法 17.1 的处理相同. 如果第三个参数 $Plot$ 是 1, 程序将画出底部的形状和质心的位置. 变量 EIco 必须预先给定. 程序返回 T, θ 以及初始底部面积的大小 So. 下一节将说明如何使用这些程序.

17.6 形变物体的三维图形表示

17.6.1 中心化底部

一个底部为三角形的例子将说明 Body 程序的用法而不必调用 MsCe 程序.

```
> restart;
> read(`Tri.sav`):
> a := 8:  b := 7:  c := 6:
> Tr := [unapply(r1, phi), unapply(r2, phi), unapply(r3, phi)];
```

$$Tr := [\phi \to -\frac{4}{21}\frac{\sqrt{132}}{\sin(\phi)}, \phi \to 16\frac{\sqrt{132}}{50\sin(\phi)+8\sqrt{132}\cos(\phi)},$$
$$\phi \to -16\frac{\sqrt{132}}{-34\sin(\phi)+8\sqrt{132}\cos(\phi)}]$$

```
> TrBo := [-Pi + Aa, -Ab, Ac, Pi + Aa];
```

$$TrBo := [-\pi + \arctan(\frac{4}{59}\sqrt{132}), -\arctan(\frac{4}{67}\sqrt{132}), \frac{1}{2}\pi + \arctan(\frac{1}{132}\sqrt{132}), \pi + \arctan(\frac{4}{59}\sqrt{132})]$$

```
> So := simplify(Sb);
```

$$So := \frac{21}{4}\sqrt{15}$$

```
> T := 0:  theta := 0:  n:= nops(Tr):  EIco := 12:
```

对于三角形不需要用数值积分的变量 EIco, 因为积分可以解析地计算.

算法 17.2 计算质心

```
MsCe := proc (rho::list, bords::list, Plot)
  local i, Mx, My, Tx, Ty, PT, pl;
  global So, T, theta;
  So := 0;
  Mx := 0;
  My := 0;
  for i to n do;
     So := So + evalf(Int(1/2*rho(u)[i]^2,
                    u = bords[i] .. bords[i + 1], EIco));
     Mx := Mx + evalf(Int(1/3*rho(u)[i]^3*cos(u),
                    u = bords[i] .. bords[i + 1], EIco));
     My := My + evalf(Int(1/3*rho(u)[i]^3*sin(u),
                    u = bords[i] .. bords[i+1], EIco));
  od;
  Tx := Mx/So;
  Ty := My/So;
  readlib(polar);
  PT := polar(Tx + (-1)^(1/2)*Ty);
  T := op(1, PT);
  theta := op(2, PT);
  if T < 1/10000 then
     T := 0;
     theta := 0;
  fi;
```

```
    if abs(theta) < 1/10000 then theta := 0; fi;
    if Plot = 1 then
      for i to n do
        pl[i] := plots[polarplot](rho(u)[i],
                        u = bords[i] .. bords[i + 1]);
      od;
      pl[0] := plot([[Tx, Ty]], style = point, symbol = box);
      pl[-1] := plot([[Tx, Ty], [0, 0]]);
      print(plots[display]({seq(pl[i], i = -1 .. n)}));
    fi;
    print('So = ',So,'   T = ',T,'     ','theta',' = ',theta);
  end;
```

```
> read('Body.map'):
> with(plots):
> for i from 0 to 3 do;
>   h := 16 - 4*i;
>   Body(Tr, TrBo, 16, h, 0.4);
>   for j from 1 to n do:
>     PL[j] := cylinderplot(Rho(u, z)[j], u = TrBo[j]..TrBo[j + 1],
>             z = 0..h, grid=[3^i + 1, 3^i + 1]);
>   od:
>   SPL[i] := display({seq(PL[j], j = 1..n)}, color = black,
>             style = hidden):
> od:
```

To display the compressed body use the Maple command :

$$cylinderplot(r(f,z),\ f = a..b,\ z = 0..h)$$

```
> display({seq(SPL[i], i = 0..3)}, axes=boxed,
>           scaling = constrained);
```

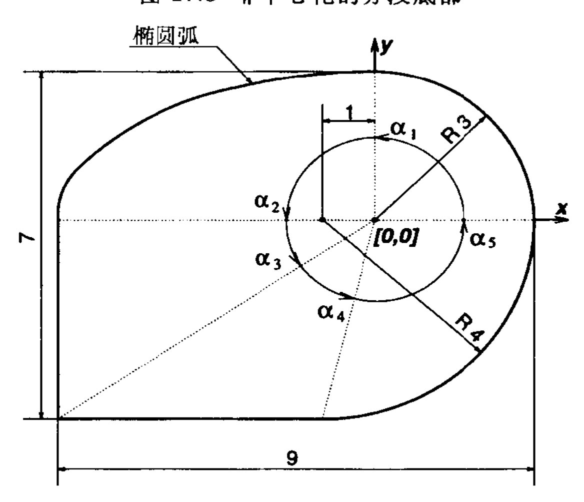
结果显示在图 17.9 中

17.6.2 非中心化的分段底部

现在讨论如图 17.8 描述的更复杂的底部，(尽管在工艺上不一定有相应的情况)

```
> restart;
> EL := 18/sqrt(36*sin(u)^2 + 9*cos(u)^2):
> p1 := -6/cos(u):
> p2 := -4/sin(u):
> k2 := solve(subs(x = r*cos(u), y = r*sin(u),
>           (x+1)^2 + y^2 = 16) ,r)[1]:
> CC := [u -> 3, unapply(EL, u), unapply(p1, u), unapply(p2, u),
>           unapply(k2, u)];
```

图 17.8 非中心化的分段底部



$$CC := [3,\ u \to \frac{6}{\sqrt{4\sin(u)^2+\cos(u)^2}},\ u \to -\frac{6}{\cos(u)},\ u \to -\frac{4}{\sin(u)},\ u \to -\cos(u)+\sqrt{\cos(u)^2+15}]$$

```
> alpha[3] := Pi + arctan(4/6):
> alpha[4] := 3/2*Pi - arctan(1/4):
> CB := [0, Pi/2, Pi, alpha[3], alpha[4], 2*Pi];
```

$$CB := [0,\ \frac{1}{2}\pi,\ \pi,\ \pi+\arctan(\frac{2}{3}),\ \frac{3}{2}\pi-\arctan(\frac{1}{4}),\ 2\pi]$$

```
> save(CC,CB, 'crazy.sav');
> EIco := 10, _CCquad:
> n := nops(CC):
> read('MsCe.map'):
> MsCe(CC, CB, 0);
```

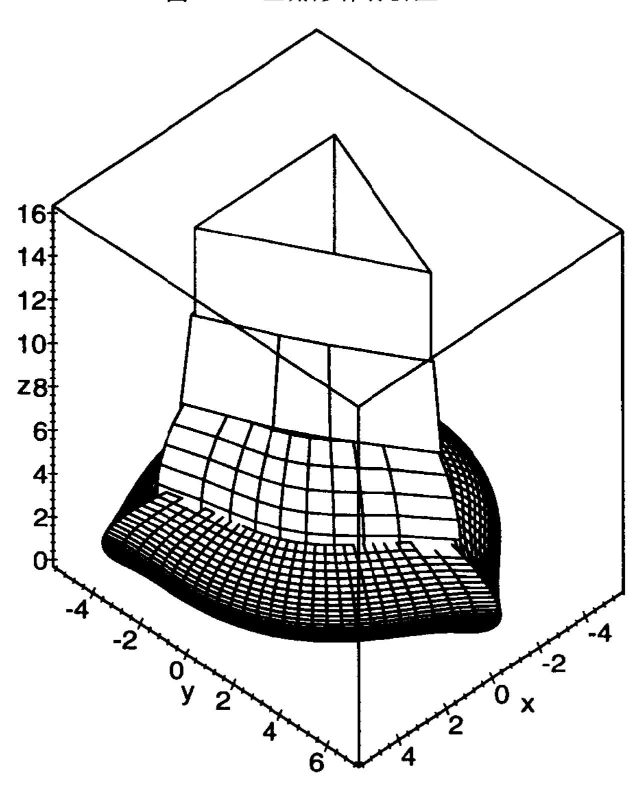
So = , 53.77212102, T = , 1.760718963, , θ, = , −2.770499706

```
> read('Body.map'):
> Body(CC, CB, 20, 8, 0.2);
```

To display the compressed body use the Maple command :
plot3d([x(f,z), y(f,z), z], f = a..b, z = 0..h)

```
> with(plots):
> for i from 1 to n do:
>   pl[i] := plot3d([X(u, z)[i], Y(u, z)[i], z],
>            u = CB[i]..CB[i + 1], z = 0..8);
> od:
> display({seq(pl[i], i = 1..n)}, color = black,
>         style = hidden, orientation = [-100, 15], axes = normal);
```

图 17.9 三角形棒材挤压



用程序 MsCe 和 Body 很容易将结果直观地用图形显示. 被挤压的物体显示在图 17.10 中.

17.6.3 凸多边形底部

在 17.3 和 17.6 节中, 已经看到如何压缩一个三角形的棒材. 三角形用它的边来描述. 用各个顶点来描述的多边形需要将算法稍微修改一下. 下面 MAPLE 的例子表示了五个顶点的解. 并没有对顶点数目的限制, 从多边形的质心看, 要按反时针顺序对这些顶点排序.

```
> restart;
> PNTi := [[1,1], [4,3], [3,4], [2,4], [0,2]]:
> PNTo := [PNTi[], PNTi[1]]:
> n := nops(PNTi):
> Tx := sum(op(i, PNTi)[1], i=1..n)/n;
> Ty := sum(op(i, PNTi)[2], i= 1..n)/n;
```

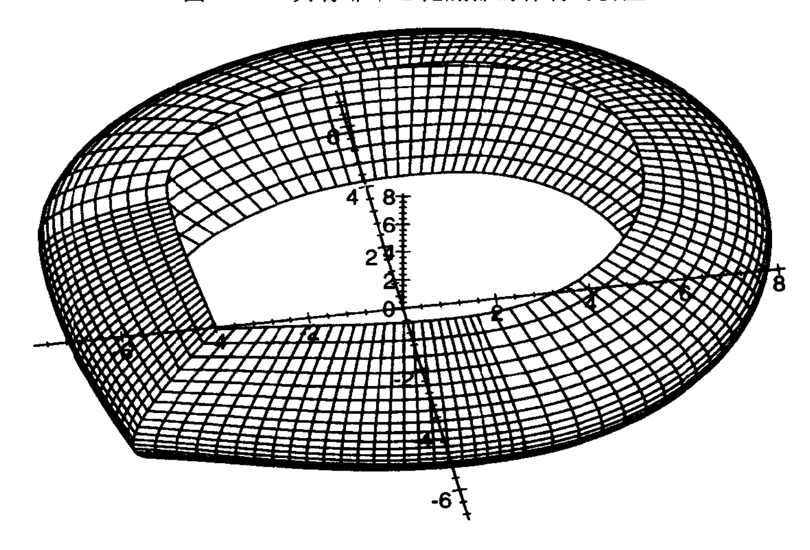
$$Tx := 2$$
$$Ty := \frac{14}{5}$$

顶点的质心用于计算多边形的质心. 我们将用原点在 $[T_x, T_y]$ 的新坐标系来描述这个多边形.

```
> readlib(polar): read(`MsCe.map`): EIco := 10, _CCquad:
> T := op(1, polar(Tx + I*Ty)): theta := op(2, polar(Tx + I*Ty)):
> while evalf(T) > 0 do;
```



图 17.10 具有非中心化底部的棒材的挤压




```
>    Tx := T*cos(theta):
>    Ty := T*sin(theta):
>    PNTo := [seq([op(i, PNTo)[1] - Tx, op(i, PNTo)[2] - Ty],
>              i = 1..n+1)];
>    i := 'i'; j := i + 1;
>    r := [seq(unapply(normal((PNTo[j, 1]*PNTo[i, 2]
>          -PNTo[i, 1]*PNTo[j, 2])/
>           (sin(u)*(PNTo[j, 1] - PNTo[i, 1]) +
>           cos(u)*(PNTo[i, 2] - PNTo[j, 2]))), u), i = 1..n)];
>    bo := [seq(op(2, polar(PNTo[i, 1] + I*PNTo[i,2 ])),
>            i = 1..n + 1)]:
>    for i from 1 to n do;
>       if evalf(bo[j]) < evalf(bo[i]) then
>          bo[j] := bo[j] + 2*Pi;
>       fi; od:
>    MsCe(r, bo, 0);
>  od:
```

$$So\ = , 6.000000000, \quad T\ = , .1704206850, \quad , \theta, \ = , -1.902855794$$
$$So\ = , 6.000000001, \quad T\ = , 0, \quad , \theta, \ = , 0$$

```
>  read(`Body.map`):
>  Body(r, bo, 4, 2, 0.3):
```

To display the compressed body use the Maple command :
cylinderplot$(r(f, z),\ f = a..b,\ z = 0..h)$

```
>  with(plots):
>  for i from 1 to n do;
>    pl[i] := cylinderplot(Rho(u, z)[i], u=bo[i]..bo[j], z = 0..2);
```

```
> od:
> display({seq(pl[i], i = 1..n)});
```

被挤压的物体的没有显示在这里，它可以象前面一样被画出.

17.7 三维动画

如果我们的计算机有足够的内存，可以把整个挤压过程象电影一样显示. 考虑一个底部如图17.8 所描述的棒材的挤压过程. 我们将再次使用已经计算好的存储在文件 `crazy.sav` 中的变量. MAPLE 的命令是相同的，但我们在不同的时间段调用程序 `Body`，从而得到挤压物体的图形序列.

```
> restart;
> Grid := [5, 5, 3, 3, 5]:
> Frames := 20:  Hi := 20:  hf := 6:  k := 1/3:
> read('crazy.sav'):  read('MsCe.map'):
> n := nops(CC): EIco := 10,_Dexp:
> MsCe(CC,CB,1):
```

$$So\ =\ ,53.77212102,\quad T\ =\ ,1.760718963,\quad ,\theta,\ =\ ,-2.770499706$$

```
> read('Body.map'):
> for j from 0 to Frames do;
>   h := Hi - (Hi - hf)*j/Frames;
>   Body(CC, CB, Hi, h, k);
>   An[j] := [seq(op(plot3d([X(u, z)[i], Y(u, z)[i], z],
>             u = CB[i]..CB[i+1], z = 0..h, grid = [Grid[i], 15])),
>             i = 1..n)];
>   print(j);
> od:
```

⋮ ⋮ ⋮

To display the compressed body use the Maple command :
plot3d$([x(f,z),\ y(f,z),\ z],\ f=a..b,\ z=0..h)$
20

为了节约篇幅，没有对每一帧打印所有 20 个相似的输出. 这些命令建立了 21 个部分图，（参见图17.11)，用它们来产生动画的数据. 在 `An` 中汇集的片段图最后形成了动画.

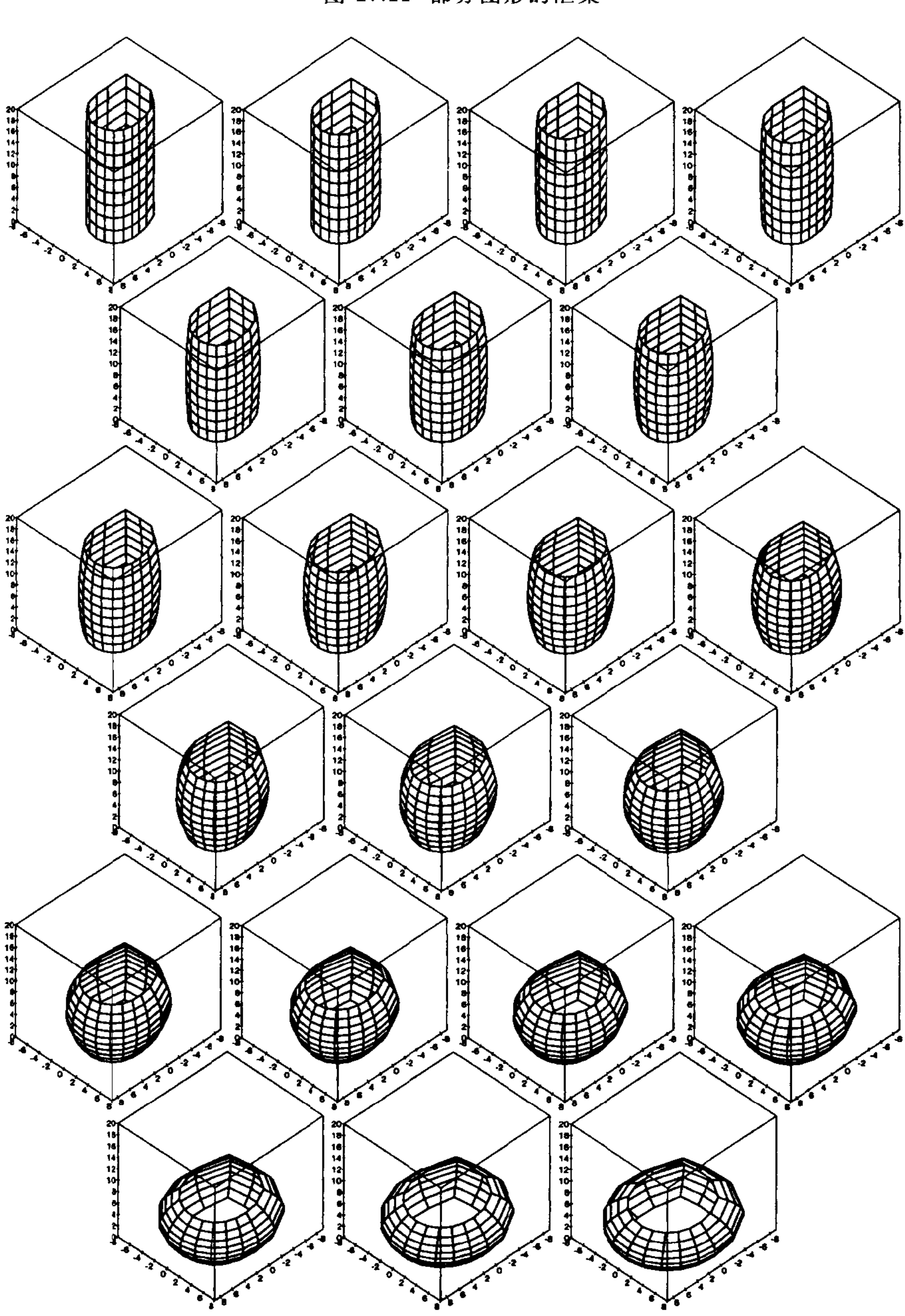
```
> PLOT3D(ANIMATE(seq(An[i], i = 0..Frames)),TITLE('Animation'),
>                   AXESLABELS(x,y,z));
```

17.8 局限性和结论

我们的模型仅仅描述了被挤压物体侧面的几何形状. 因此它不能描述物体材料的移动，并且局限于描述凸边形的横截面. 然而这个简单模型很适于描述均匀的和物理上各向同性（尤其是受压特性）的材料. 模型的精度取决于比例数 b/a，其中 b 和 a 分别是棒材初始横截面的最短直径和最长直径.

金属物体形变的这个数学模型可以成功地用于工艺计算，使工艺实验的次数大大减少，从而节约时间和金钱.

图 17.11 部分图形的汇集



参考文献

[1] S. BARTOŇ, *Free Metal Compression*, in Solving Problems in Scientific Computing using MAPLE and MATLAB, W. Gander and J. Hřebíček, editors, Springer, 1993, 1995.

[2] B. AVITZUR, *Metal Forming*, Marcel Dekker, New York, 1980.

[3] T. Z. BLAZYNSKI, *Metal Forming*, MacMillan Press LTD, London, 1976.

[4] A. FARLÍK AND E. ONDRÁČEK, *Theory of the dynamic forming (in czech)*, SNTL, Praha, 1968.

[5] S. R. REID, *Metal Forming*, Pergamon Press, Oxford, 1985.

第十八章　Gauss 积分

U. von Matt

18.1　引言

在本章中，我们借助 MAPLE 研究怎样计算 Gauss 积分．考虑一下积分

$$\int_a^b f(x)\omega(x)\mathrm{d}x, \tag{18.1}$$

其中 $\omega(x)$ 是一个非负权函数．假设对所有的 $k \geq 0$, 积分

$$\int_a^b |x|^k \omega(x)\mathrm{d}x \tag{18.2}$$

存在．还假设 $\omega(x)$ 在区间 $[a,b]$ 内只有有限个零点 (参见 [20, 第 18 页]).

Gauss 积分的目的是用有限和

$$\int_a^b f(x)\omega(x)\mathrm{d}x \approx \sum_{i=1}^{m+n} w_i f(x_i) \tag{18.3}$$

来近似积分 (18.1), 其中需要确定横坐标 x_i 和权 w_i, 使得所有不超过最高可能次数的多项式都精确可积.

我们还要考虑 Gauss-Radau 积分和 Gauss-Lobatto 积分，其中规定在区间 $[a,b]$ 的边界上有 m 个横坐标，用 $x_1, x_2, \ldots, x_m$ 表示这 m 个横坐标， m 可取 $m=0, m=1$ 和 $m=2$.

Gauss 积分理论是一个广泛的领域．在本章里，我们将集中在导出积分法则的部分．读者也可参考有关的文献 [4, 20, 31].

在 Gauss 积分中，正交多项式起了关键作用．只要知道正交多项式的递推关系式，利用一个特征值分解，就可以计算 Gauss 积分法则的横坐标和权．因此在第 18.2 节，我们将导出计算正交多项式的两种方法.

在第一种情况, 区间 $[a,b]$ 和权函数 $\omega(x)$ 必须是显式的. 于是利用符号积分, 我们可以用 Lanczos 算法计算正交多项式的递推关系式.

在其它情况，只需要求出对应于区间 $[a,b]$ 和权函数 $\omega(x)$ 的矩．基于 Gram 矩阵的 Cholesky 分解，我们提出了一个算法，计算正交多项式的递推关系式.

在第 18.3 节，我们证明一个定理，它是计算 Gauss 积分法则的基础．在此定理的基础上，我们分别在第 18.4 节，第 18.5 节和第 18.6 节，给出计算 Gauss,Gauss-Radau 和 Guass-Lobatto 积分法则的算法．权 w_i 的计算是在第 18.7 节．最后，在第 18.8 节，我们得到积分误差的一个表达式.

计算 Gauss 积分法则是 MAPLE 一个有价值的应用．我们利用它的符号能力，精确地计算递推关系式的系数．因为 MAPLE 也支持任意精度的浮点运算，所以对任何要求的精度，利用特征值分解，我们均可计算积分法则的横坐标和权.

18.2 正交多项式

在本节，我们讨论计算对应于区间 $[a,b]$ 和权函数 $\omega(x)$ 的正交多项式的几种方法.

定义 18.0.1 我们称

$$(f,g) := \int_a^b f(x)g(x)\omega(x)\mathrm{d}x \tag{18.4}$$

是关于权函数 $\omega(x)$ 和区间 $[a,b]$ 的内积.

此内积有一个重要的性质

$$(xf,g) = (f,xg), \tag{18.5}$$

在本章中，经常用到此性质.

定义 18.0.2 我们称表达式

$$\mu_k := (x^k, 1) = \int_a^b x^k\omega(x)\mathrm{d}x \tag{18.6}$$

是关于权函数 $\omega(x)$ 的第 k 个矩.

很多书是关于从这些常矩计算积分法则的 [1, 6, 28, 33]. 然而，这种方法是数值不稳定的，正如 [9] 中 Gautschi 所证明的. 作为一种改进，引进所谓修正矩 (参见 [10, 第 245 页] 和 [27, 第 466 页]).

定义 18.0.3 我们称表达式

$$\upsilon_k := (\pi_k, 1) = \int_a^b \pi_k(x)\omega(x)\mathrm{d}x \tag{18.7}$$

是关于权函数 $\omega(x)$ 的第 k 个修正矩，其中 π_k 是一个 k 次的多项式.

假设多项式 $\pi_0(x), \pi_1(x), \ldots$ 满足递推关系式

$$x\pi_{k-1} = \hat{\gamma}_{k-1}\pi_{k-2} + \hat{\alpha}_k\pi_{k-1} + \hat{\beta}_k\pi_k, \quad k = 1, 2, \ldots \tag{18.8}$$

其中 $\hat{\gamma}_0 := 0$ 和 $\pi_{-1}(x) := 0$. 假设系数 $\hat{\beta}_k$ 是非零的.

常矩 μ_k 是修正矩 υ_k 的一个特例，令 $\pi_k(x) = x^k$ 即可得到，在这种情况下，递推关系式 (18.8) 中， $\hat{\alpha}_k = \hat{\gamma}_k = 0, \hat{\beta}_k = 1$.

定义 18.0.4 我们定义矩阵

$$M := \begin{bmatrix} m_{00} & \cdots & \cdots & m_{0,n-1} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ m_{n-1,0} & \cdots & \cdots & m_{n-1,n-1} \end{bmatrix} \tag{18.9}$$

是 n 阶 *Gram* 矩阵，其中每个元素

$$m_{ij} := (\pi_i, \pi_j) \tag{18.10}$$

是一个内积.

矩阵 M 不仅是对称的，而且对 $\mathbf{v} \neq \mathbf{0}$, 由于

$$\mathbf{v}^{\mathrm{T}}M\mathbf{v} = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} v_i m_{ij} v_j = (\sum_{i=0}^{n-1} v_i\pi_i, \sum_{j=0}^{n-1} v_j\pi_j) > 0,$$

它也是正定的.

现在我们说明, 从修正矩 v_k 和递推关系式 (18.8), 如何计算矩阵 M. 在 [10, 第 255-256 页] 中有一个类似的导出结果.

因为 $\pi_0(x)$ 是一个常数, 所以有

$$m_{i0} = (\pi_i, \pi_0) = \pi_0 v_i, \quad m_{0j} = (\pi_0, \pi_j) = \pi_0 v_j. \tag{18.11}$$

因而在下面, 我们将假设 $i > 0$ 和 $j > 0$. 如果用递推关系式 (18.8) 代替 (18.10) 中的多项式 $\pi_j(x)$, 则得到

$$\begin{aligned} m_{ij} &= (\pi_i, \pi_j) = \tfrac{1}{\hat{\beta}_j}(\pi_i, x\pi_{j-1} - \hat{\gamma}_{j-1}\pi_{j-2} - \hat{\alpha}_j\pi_{j-1}) \\ &= \tfrac{1}{\hat{\beta}_j}((x\pi_i, \pi_{j-1}) - \hat{\gamma}_{j-1}m_{i,j-2} - \hat{\alpha}_j m_{i,j-1}). \end{aligned}$$

为了代替 $x\pi_i(x)$, 应用几次递推关系式 (18.8), 可得

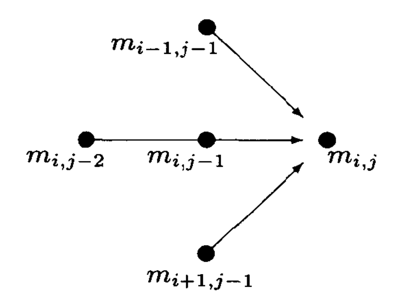
$$m_{ij} = \frac{1}{\hat{\beta}_j}((\hat{\gamma}_i\pi_{i-1} + \hat{\alpha}_{i+1}\pi_i + \hat{\beta}_{i+1}\pi_{i+1}, \pi_{j-1}) - \hat{\gamma}_{j-1}m_{i,j-2} - \hat{\alpha}_j m_{i,j-1})$$

或

$$m_{ij} = \frac{1}{\hat{\beta}_j}(\hat{\gamma}_i m_{i-1,j-1} + (\hat{\alpha}_{i+1} - \hat{\alpha}_j)m_{i,j-1} + \hat{\beta}_{i+1}m_{i+1,j-1} - \hat{\gamma}_{j-1}m_{i,j-2}), \tag{18.12}$$

其中 $m_{i,-1} = 0$. 因而由初值 (18.11) 开始, 在 (18.12) 中可递推地建立矩阵 M.

下列模型也可描述 (18.12) 中的关系:



此图说明, 计算 m_{ij} 的值之前, 必须知道矩阵 M 的元素. 用 MAPLE 计算 Gram 矩阵 M 的方法见算法 18.1.

定义 18.0.5 *如果下列条件成立:*

1. *$p_k(x)$ 是 k 次的,*
2. *对于次数小于 k 的所有多项式 $p(x)$, 多项式 $p_k(x)$ 满足正交条件*

$$(p_k, p) = 0, \tag{18.13}$$

3. *多项式 $p_k(x)$ 满足归一化条件*

$$(p_k, p_k) = 1, \tag{18.14}$$

则称多项式 $p_0(x), p_1(x), \ldots$ 是关于内积 (18.4) 的正交多项式.

由于条件 (18.14), 多项式 p_0 一定是不为零的常数. 因此, (18.13) 的一个简单推论是, 对于 $k \geq 1$

$$(p_k, 1) = 0. \tag{18.15}$$

下面的定理保证了关于内积 (18.4) 正交多项式的存在性, 并且显示了计算它们的第一种方法.

算法 18.1 从改进的矩来计算 Gram 矩阵 M

```
Gram := proc (pi0, alphahat, betahat, gammahat, nu, n)
  local i, j, M;

  M := array (symmetric, 0..2*n-1, 0..2*n-1);
  for i from 0 to 2*n-1 do
    M[i,0] := simplify (pi0 * nu[i]);
  od;
  for i from 1 to 2*n-2 do
    M[i,1] := simplify (
               (gammahat[i] * M[i-1,0] +
                (alphahat[i+1] - alphahat[1]) * M[i,0] +
                betahat[i+1] * M[i+1,0]) / betahat[1]);
  od;
  for j from 2 to n-1 do
    for i from j to 2*n-1-j do
      M[i,j] := simplify (
                 (gammahat[i] * M[i-1,j-1] +
                  (alphahat[i+1] - alphahat[j]) * M[i,j-1] +
                  betahat[i+1] * M[i+1,j-1] -
                  gammahat[j-1] * M[i,j-2]) / betahat[j]);
    od;
  od;

  RETURN (M);
end:
```

定理 18.1 *对于任意给定的容许内积，存在一个正交多项式序列* $\{p_k(x)\}_{k=0}^{\infty}$.

证明 根据 Mysovskih[22] 的论文，设 $p_k(x)$ 为

$$p_k(x) = \sum_{i=0}^{k} s_{ki}\pi_i(x). \tag{18.16}$$

我们定义下三角矩阵

$$S := \begin{bmatrix} s_{00} & & & \\ \vdots & \ddots & & \\ \vdots & & \ddots & \\ s_{n-1,0} & \cdots & \cdots & s_{n-1,n-1} \end{bmatrix}. \tag{18.17}$$

对于 $k \neq l$, 正交性条件 (18.13) 隐含:

$$(\sum_{i=0}^{k} s_{ki}\pi_i, \sum_{j=0}^{l} s_{lj}\pi_j) = \sum_{i=0}^{k}\sum_{j=0}^{l} s_{ki}s_{lj}m_{ij} = 0. \tag{18.18}$$

归一化条件 (18.14) 即为

$$(\sum_{i=0}^{k} s_{ki}\pi_i, \sum_{i=0}^{k} s_{ki}\pi_i) = \sum_{i=0}^{k}\sum_{j=0}^{k} s_{ki}s_{kj}m_{ij} = 1. \tag{18.19}$$

可用矩阵表示条件 (18.18) 和 (18.19)

$$SMS^{\mathrm{T}} = I,$$

其中 M 是 (18.9) 的 Gram 矩阵. 因为 M 是一个对称正定矩阵, 所以可计算 Cholesky 分解

$$M = LL^{\mathrm{T}}, \tag{18.20}$$

其中 L 是下三角矩阵. 由于 L 的逆矩阵也是一个下三角矩阵, 因此找到所求的矩阵 S,$S := L^{-1}$.

与 (18.16) 相反, 正交多项式的下列表示经常更为有用.

定理 18.2 正交多项式 $p_0(x), p_1(x), \ldots$ 满足递推关系式

$$xp_{k-1} = \beta_{k-1}p_{k-2} + \alpha_k p_{k-1} + \beta_k p_k, \quad k = 1, 2, \ldots \tag{18.21}$$

其中

$$\alpha_k = (xp_{k-1}, p_{k-1}) \tag{18.22}$$

$$\beta_k = (xp_{k-1}, p_k), \tag{18.23}$$

并且 $\beta_0 := 0, p_{-1}(x) := 0$, 和 $p_0(x) := \pm\sqrt{1/\mu_0} = \pm\sqrt{\pi_0/v_0}$.

证明 显然, xp_{k-1} 可写成

$$xp_{k-1} = \sum_{i=0}^{k} c_i p_i$$

其中 $c_i = (p_i, xp_{k-1}) = (xp_i, p_{k-1})$. 由于 (18.13), 我们有 $c_0 = \cdots = c_{k-3} = 0$. 如果根据 (18.22) 和 (18.23) 定义 α_k 和 β_k, 可立即得到 (18.21).

此定理得到第一个计算正交多项式的递归算法. 如果已知系数 $\alpha_1, \ldots, \alpha_k$ 和 $\beta_1, \ldots, \beta_{k-1}$, 以及多项式 $p_0, \ldots, p_{k-1}$, 则由方程 (18.21), 就可确定 p_k. 我们有

$$q_k := \beta_k p_k = (x - \alpha_k)p_{k-1} - \beta_{k-1}p_{k-2}. \tag{18.24}$$

标准化条件 (18.14) 隐含

$$\beta_k = \pm\sqrt{(q_k, q_k)}, \tag{18.25}$$

于是 $p_k = q_k/\beta_k$. 最后, 由 (18.22) 可得 α_{k+1} 的值.

算法 18.2 Lanczos 算法

```
Lanczos := proc (iprod, alpha, beta, n)
  local k, p, q, x;

  alpha := array (1..n);
  if n > 1 then beta := array (1..n-1) else beta := 'beta' fi;

  p[0] := collect (1 / sqrt (iprod (1, 1, x)), x, simplify);
  alpha[1] := normal (iprod (x*p[0], p[0], x));
  q := collect ((x - alpha[1]) * p[0], x, simplify);
  for k from 2 to n do
    beta[k-1] := normal (sqrt (iprod (q, q, x)));
    p[k-1] := collect (q / beta[k-1], x, simplify);
    alpha[k] := normal (iprod (x*p[k-1], p[k-1], x));
    q := collect ((x - alpha[k]) * p[k-1] - beta[k-1]*p[k-2],
                  x, simplify);
  od;

  RETURN (NULL);
end:
```

这个递归程序也称为 Lanczos 算法. 算法 18.2 给出它在 MAPLE 中的实现. 注意, 内积 (18.4) 作为参数 iprod 被传递. 例如, 我们用语句

```
> a := 0: b := infinity:
> omega := t -> exp(-t):
> iprod := (f,g,x) -> int(f*g*omega(x),x=a..b):
> n := 10:
> Lanczos(iprod, 'alpha', 'beta', n):
```

计算 Laguerre 多项式的递归系数 $\alpha_k = 2k-1$ 和 $\beta_k = k$.

Lanczos 算法要求计算任意多项式内积 (18.4) 的能力. 仅基于修正矩 (18.7), 也可得到另一个算法.

在定理 18.1 的证明中, 我们提出一种计算正交多项式的方法, 它是基于 Gram 矩阵 M 的 Cholesky 分解 (18.20). 在 (18.16) 中表示 p_k 的矩阵 S 是 Cholesky 因子 L 的逆矩阵. 直接计算 Cholesky 分解 (18.20), 需要 $O(n^3)$ 的运算 [17, 第 4.2 节]. 然而, 我们将证明, 矩阵 L 的元素满足一个类似 Gram 矩阵 M 的递推关系式. 这使得我们可设计一个改进的算法, 它只要求 $O(n^2)$ 的运算.

因为 $S = L^{-1}$, 由 (18.20) 可得

$$L = MS^{\mathrm{T}}.$$

于是， L 的元素 l_{ij} 可写成

$$l_{ij}=\sum_{k=0}^{j} m_{ik}s_{jk}=(\pi_i,\sum_{k=0}^{j} s_{jk}\pi_k),$$

由 (18.16) 可得

$$l_{ij}=(\pi_i,p_j). \tag{18.26}$$

由于 (18.20), 可用

$$l_{00}=\sqrt{m_{00}} \tag{18.27}$$

得出 l_{00} 的值. 在下面将假设 $i>0$. 如果利用递推关系式 (18.8), 代替 (18.26) 中的多项式 π_i, 则有

$$\begin{aligned} l_{ij} &= (\pi_i,p_j)=\tfrac{1}{\hat\beta_i}(x\pi_{i-1}-\hat\gamma_{i-1}\pi_{i-2}-\hat\alpha_i\pi_{i-1},p_j)\\ &= \tfrac{1}{\hat\beta_i}((x\pi_{i-1},p_j)-\hat\gamma_{i-1}l_{i-2,j}-\hat\alpha_i l_{i-1,j}). \end{aligned}$$

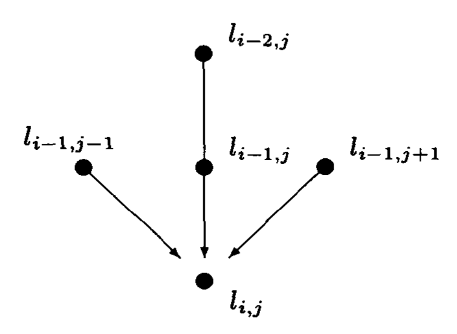
现在可用递推关系式 (18.21), 代替表达式 $(x\pi_{i-1},p_j)=(\pi_{i-1},xp_j)$ 中的多项式 $xp_j(x)$. 这样得出

$$l_{ij}=\frac{1}{\hat\beta_i}((\pi_{i-1},\beta_j p_{j-1}+\alpha_{j+1}p_j+\beta_{j+1}p_{j+1})-\hat\gamma_{i-1}l_{i-2,j}-\hat\alpha_i l_{i-1,j})$$

或

$$l_{ij}=\frac{1}{\hat\beta_i}(\beta_j l_{j-1,j-1}+(\alpha_{j+1}-\hat\alpha_i)l_{i-1,j}+\beta_{j+1}l_{i-1,j+1}-\hat\gamma_{i-1}l_{i-2,j}), \tag{18.28}$$

其中 $l_{-1,j}=0$. 我们得到矩阵 L 的元素的递推关系式. 下面的模型可描述 L 中的关系:



然而应该注意，方程 (18.28) 中有未知系数 α_k 和 β_k. 因此，根据 (18.28), 不能计算 L 中所有的元素.

如果已知 $\alpha_1,\ldots,\alpha_{k-1}$ 和 $\beta_1,\ldots,\beta_{k-1}$, 以及所有 $i<k$ 的 l_{ij}, 则可用 (18.28) 计算 l_{kj}, 对于 $j=0,\ldots,k-2$. 由 M 的 Cholesky 分解 (18.20), 可得

$$l_{k,k-1}=\frac{1}{l_{k-1,k-1}}(m_{k,k-1}-\sum_{j=0}^{k-2} l_{kj}l_{k-1,j}) \tag{18.29}$$

和

$$l_{k,k}=\sqrt{m_{k,k}-\sum_{j=0}^{k-1} l_{kj}^2} \tag{18.30}$$

的值. 另一方面，递推关系式 (18.28) 给出 $l_{k,k-1}$ 的表达式

$$l_{k,k-1}=\frac{1}{\hat\beta_k}(\beta_{k-1}l_{k-1,k-2}+(\alpha_k-\hat\alpha_k)l_{k-1,k-1}). \tag{18.31}$$

如果求解 (18.31), 则可得

$$\alpha_k = \hat{\alpha}_k + \frac{\hat{\beta}_k l_{k,k-1} - \beta_{k-1} l_{k-1,k-2}}{l_{k-1,k-1}}. \tag{18.32}$$

用同样的方法，由 (18.28) 可得

$$l_{k,k} = \frac{1}{\hat{\beta}_k} \beta_k l_{k-1,k-1}, \tag{18.33}$$

使得

$$\beta_k = \hat{\beta}_k \frac{l_{k,k}}{l_{k-1,k-1}}. \tag{18.34}$$

在 [27] 中， Sack 和 Donovan 从修正矩引进一个计算递推关系式 (18.21) 的类似方法. 他们称它为 “长商修正差分 (LQMD) 算法”. 在 [10] 中， Gautschi 提出一个相关的算法，它基于 Gram 矩阵 M 的 Cholesky 分解 (18.20). 不幸的是，他没有注意到， Cholesky 因子 L 中元素的递推关系式 (18.28). 虽然他的叙述比 Sack 和 Donovan 更为清楚，但是其算法是 $O(n^3)$ 阶的.

现在我们提出算法 18.3, 它在 MAPLE 中实现我们的方法. 对于给定的 n, 必须提供输入参数：常数多项式 π_0, 系数 $\hat{\alpha}_k, k = 1, \ldots, 2n-1$, $\hat{\beta}_k, k = 1, \ldots, 2n-1$ 和 $\hat{\gamma}_k, k = 1, \ldots, 2n-2$, 以及修正矩 $\upsilon_k, k = 0, \ldots, 2n-1$. 从而，算法 18.3 建立基于递推公式 (18.11) 和 (18.12) 的 Gram 矩阵 M. 然后，用方程 (18.27),(18.28),(18.29),(18.30),(18.32) 和 (18.34), 构造 Cholesky 因子 L, 以及递推关系式 (18.21) 的系数： $\alpha_k, k = 1, \ldots, n$ 和 $\beta_k, k = 1, \ldots, n-1$. 此算法是 $O(n^2)$ 阶的.

算法 18.3 从修正矩计算递归关系式 (18.21)

```
SackDonovan := proc (pi0, alphahat, betahat, gammahat, nu, n,
                     alpha, beta)
  local j, k, L, M;

  M := Gram (pi0, alphahat, betahat, gammahat, nu, n);
  L := array (0..n, 0..n);
  alpha := array (1..n);
  if n > 1 then beta := array (1..n-1) else beta := 'beta' fi;
  L[0,0] := simplify (sqrt (M[0,0]));
  L[1,0] := simplify (M[1,0] / L[0,0]);
  alpha[1] := simplify (
              alphahat[1] + betahat[1] * L[1,0] / L[0,0]);
  k := 1;
  while k < n do
    L[k,k] := simplify (
              sqrt (M[k,k] - sum ('L[k,j]^2', 'j'=0..k-1)));
    beta[k] := simplify (betahat[k] * L[k,k] / L[k-1,k-1]);
    k := k+1;
    L[k,0] := M[k,0] / L[0,0];
    for j from 1 to k-2 do
      L[k,j] := simplify (
```

```
                    (beta[j] * L[k-1,j-1] +
                     (alpha[j+1] - alphahat[k]) * L[k-1,j] +
                     beta[j+1] * L[k-1,j+1] -
                     gammahat[k-1] * L[k-2,j]) / betahat[k]);
      od;
      L[k,k-1] := simplify (
                    (M[k,k-1] -
                     sum ('L[k,j] * L[k-1,j]', 'j'=0..k-2)) /
                     L[k-1,k-1]);
      alpha[k] := simplify (
                    alphahat[k] +
                    (betahat[k] * L[k,k-1] -
                     beta[k-1] * L[k-1,k-2]) / L[k-1,k-1]);
    od;

    RETURN (NULL);
  end:
```

算法 18.3 所需的存储量也是 $O(n^2)$ 阶的. 正如 Sack 和 Donovan 所给出的, 可得到存储量与 n 成比例的一个更复杂的实现. 为了清楚起见, 在这方面我们将不继续下去.

读者应该注意, 算法 18.3 优于 Gautschi 所给的算法, 仅在浮点运算可达到. 在一个符号计算中, 正如在 MAPLE 中执行, 不仅运算的次数, 而且运算对象的大小都影响总的执行时间. 虽然算法 18.3 比 Gautschi 所给的算法, 需要更少的算术运算, 但是它的运算对象更为复杂. 所以我们没有看到, 我们的算法比 Gautschi 所给的算法有明显的加速.

作为算法 18.3 的一个应用例子, 我们计算关于区间 $[0,\infty]$ 和两个权函数

$$\omega_{\cos}(x) := \mathrm{e}^{-x}(1+\cos\varphi x), \tag{18.35}$$

$$\omega_{\sin}(x) := \mathrm{e}^{-x}(1+\sin\varphi x). \tag{18.36}$$

的正交多项式. Laguerre 多项式 $L_n(x)$ 是著名的, 它是关于区间 $[0,\infty]$ 和权函数 $\omega(x)=\mathrm{e}^{-x}$ 的正交多项式. 因此, 考虑改进的矩

$$v_n^{\cos} := \int_0^\infty L_n(x)\omega_{\cos}(x)\mathrm{d}x, \tag{18.37}$$

$$v_n^{\sin} := \int_0^\infty L_n(x)\omega_{\sin}(x)\mathrm{d}x. \tag{18.38}$$

是合理的. 现在我们说明, 怎样解析地计算改进的矩 $v_n^{\cos}$ 和 $v_n^{\sin}$.

定理 18.3 *多项式*

$$L_n(x) := \sum_{k=0}^{n}(-1)^{n-k}\binom{n}{k}\frac{x^k}{k!} \tag{18.39}$$

是第 n 个 Laguerre 多项式.

证明 Laguerre 多项式满足递推关系式 (18.21), 其中 $p_0=1,\alpha_k=2k-1,\beta_k=k$. 因为 (18.39) 所定义的多项式, 满足同样的递推关系式, 所以它们一定与 Laguerre 多项式相同.

对任意的复常数 c, 其中 $\mathbb{R}(c) < 0$, 我们有

$$\int_0^\infty x^n e^{cx} dx = n!(\frac{-1}{c})^{n+1}. \tag{18.40}$$

由分部积分和关于 n 的归纳法, 可证明此式. 因为方程 (18.39) 和 (18.40), 所以可证明

$$\int_0^\infty L_n(x) e^{cx} dx = (-1)^{n+1} \frac{(c+1)^n}{c^{n+1}}. \tag{18.41}$$

从而有

$$\begin{aligned}
\int_0^\infty L_n(x) e^{-x} \cos\varphi x dx &= \mathbb{R}(\int_0^\infty L_n(x) e^{(-1+i\varphi)x} dx) \\
&= (-1)^{n+1} \mathbb{R}(\tfrac{(i\varphi)^n}{(-1+i\varphi)^{n+1}}) \\
&= \tfrac{(-1)^n \varphi^n}{(1+\varphi^2)^{n+1}} \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n+1}{2k+1} (-1)^k \varphi^{n-2k}
\end{aligned}$$

和

$$\begin{aligned}
\int_0^\infty L_n(x) e^{-x} \sin\varphi x dx &= \Im(\int_0^\infty L_n(x) e^{(-1+i\varphi)x} dx) \\
&= (-1)^{n+1} \Im(\tfrac{(i\varphi)^n}{(-1+i\varphi)^{n+1}}) \\
&= \tfrac{(-1)^n \varphi^n}{(1+\varphi^2)^{n+1}} \sum_{k=0}^{\lfloor \frac{n+1}{2} \rfloor} \binom{n+1}{2k} (-1)^k \varphi^{n+1-2k}.
\end{aligned}$$

现在，我们可用

$$\nu_0^{\cos} = 1 + \frac{1}{1+\varphi^2}, \tag{18.42}$$

$$\nu_n^{\cos} = \frac{(-1)^n \varphi^n}{(1+\varphi^2)^{n+1}} \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n+1}{2k+1} (-1)^k \varphi^{n-2k}, \quad n > 0, \tag{18.43}$$

$$\nu_0^{\sin} = 1 + \frac{\varphi}{1+\varphi^2}, \tag{18.44}$$

$$\nu_n^{\sin} = \frac{(-1)^n \varphi^n}{(1+\varphi^2)^{n+1}} \sum_{k=0}^{\lfloor \frac{n+1}{2} \rfloor} \binom{n+1}{2k} (-1)^k \varphi^{n+1-2k}, \quad n > 0, \tag{18.45}$$

表示修正矩 (18.37) 和 (18.38). 利用 MAPLE 语句

```
> N := 10:
> phi :=1:

> alphahat :=array (1..2*N-1, [seq (2*k-1, k=1..2*N-1)]):
> betahat := array (1..2*N-1, [seq (k, k=1..2*N-1)]):
> gammahat := array (1..2*N-2, [seq (k, k=1..2*N-2)]):
> L := (n, x) -> sum('(-1)^(n-k)*binomial(n,k)* x^k / k!',
>                    'k'=0..n):
> nucos[0] := 1 + 1 /(1+phi^2):
> for n from 1 to 2*N-1 do
>   nusin[n] := (-1)^n * phi^n / (1+phi^2)^(n+1) *
```

```
>               sum('binomial (n+1,2*k+1) *
>                  (-1)^k*phi^(n-2*k)','k'=0..floor(n/2));
> od:
> nusin[0] := 1 + phi / (1+phi^2):
> for n from 1 to 2*N-1 do
>   nusin[n] := (-1)^n * phi^n / (1+phi^2)^(n+1) *
>               sum('binomial (n+1,2*k) * (-1)^k *
>                  phi^(n+1-2*k)','k'=0..floor((n+1)/2));
> od:

> SackDonovan (L (0, x), alphahat, betahat, gammahat,
>               nucos, N, 'alphacos', 'betacos');
> SackDonovan (L (0, x), alphahat, betahat, gammahat,
>               nusin, N, 'alphasin', 'betasin');
```

表 18.1 $\varphi=1$ 时 $\alpha_n^{\cos}$ 和 $\beta_n^{\cos}$ 的值

n	$\alpha_n^{\cos}$	$\beta_n^{\cos}$
1	$\frac{2}{3}\approx 0.66667$	$\frac{\sqrt{5}}{3}\approx 0.74536$
2	$\frac{53}{15}\approx 3.5333$	$\frac{12}{5}\approx 2.4000$
3	$\frac{329}{80}\approx 4.1125$	$\frac{\sqrt{1655}}{16}\approx 2.5426$
4	$\frac{4489}{5296}\approx 8.4005$	$\frac{8\sqrt{29499}}{331}\approx 4.1511$
5	$\frac{26238754}{3254723}\approx 8.0617$	$\frac{5\sqrt{91073657}}{9833}\approx 4.8527$
6	$\frac{31146028765}{2705520451}\approx 11.512$	$\frac{18\sqrt{10403284501}}{275147}\approx 6.6726$
7	$\frac{41962386991493}{3493256406708}\approx 12.012$	$\frac{7\sqrt{123806905335947}}{12695964}\approx 6.1349$
8	$\frac{99084727782033173}{5712757228305564}\approx 17.344$	$\frac{8\sqrt{213374749185568311}}{449966401}\approx 8.2126$
9	$\frac{465045058223400793942}{30249445557756103921}\approx 15.374$	$\frac{15\sqrt{158546041858245604102 9}}{67226009521}\approx 8.8845$

表 18.2 $\varphi=1$ 时 $\alpha_n^{\sin}$ 和 $\beta_n^{\sin}$ 的值

n	$\alpha_n^{\sin}$	$\beta_n^{\sin}$
1	$1\approx 1.0000$	$\frac{\sqrt{6}}{3}\approx 0.81650$
2	$\frac{5}{2}\approx 2.5000$	$\frac{\sqrt{165}}{6}\approx 2.1409$
3	$\frac{127}{22}\approx 5.7727$	$\frac{18\sqrt{65}}{55}\approx 2.6386$
4	$\frac{2578}{429}\approx 6.0093$	$\frac{\sqrt{784190}}{195}\approx 4.5413$
5	$\frac{2747911}{278031}\approx 9.8835$	$\frac{830\sqrt{1365}}{7129}\approx 4.3015$
6	$\frac{14684038321}{1375127068}\approx 10.6778$	$\frac{3\sqrt{176313765003}}{192892}\approx 6.5306$
7	$\frac{21638014309259}{1590195668348}\approx 13.607$	$\frac{16268\sqrt{11567141}}{8243969}\approx 6.7114$
8	$\frac{9020385319743068}{667514062758403}\approx 13.513$	$\frac{54\sqrt{16509262674612 3}}{80969987}\approx 8.5691$
9	$\frac{246930751927882600 1}{131341029442952049}\approx 18.801$	$\frac{\sqrt{1615594742551806361 94}}{1622095227}\approx 7.8359$

我们可计算，关于区间 $[0,\infty]$ 和两个权函数 $\omega_{\cos}$ 和 $\omega_{\sin}$ 的正交多项式的递推关系式 (18.21). 得到表 18.1 和 18.2 所示的结果.

定理 18.4 正交多项式 $p_k(x)$ 仅有 k 个不同的零点， $a<x_1<\cdots<x_k<b$.

证明 用反证法证明 (参见 [20, 第 21 页]). 假设在区间 (a,b) 内, 多项式 $p_k(x)$ 只有 $k' < k$ 个零点 ζ_i, 具有奇重数. 在这个区间内, 多项式

$$q(x) := \prod_{i=1}^{k'}(x-\zeta_i)$$

与 $p_k(x)$ 有相同的符号, 因此, 在区间 (a,b) 内, 积 $p_k(x)q(x)$ 不改变符号, 即

$$(p_k, q) \neq 0,$$

与 $p_k(x)$ 的正交性质 (18.13) 矛盾.

下面的定理, 建立了由递推关系式 (18.21) 所定义的多项式序列 $\{p_k\}_{k=0}^{n}$ 和系数为 α_k, β_k 的三对角线矩阵 T 的特征值分解之间的联系.

定理 18.5 *对于一个可由递推关系式 (18.21) 表示的 n 次多项式 $p_n(x)$(其中 $\beta_k \neq 0$), 三对角线矩阵*

$$T_n := \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{bmatrix} \tag{18.46}$$

的特征值 $x_1 < \ldots < x_n$ 是它的零点. 进一步令

$$T_n = QXQ^{\mathrm{T}} \tag{18.47}$$

是 T_n 的特征值分解, 其中 X 是对角线元素为 $x_1, \ldots, x_n$ 的特征值矩阵, Q 是正交的特征向量矩阵. 如果定义

$$P := \begin{bmatrix} p_0(x_1) & \cdots & p_0(x_n) \\ \vdots & \vdots & \vdots \\ p_{n-1}(x_1) & \cdots & p_{n-1}(x_n) \end{bmatrix} \tag{18.48}$$

和

$$W := \frac{1}{p_0}\begin{bmatrix} q_{11} & & 0 \\ & \ddots & \\ 0 & & q_{1n} \end{bmatrix} \tag{18.49}$$

则

$$Q = PW. \tag{18.50}$$

证明 可将递推关系式 (18.21) 写成矩阵形式

$$x\begin{bmatrix} p_0 \\ \vdots \\ \vdots \\ p_{n-1} \end{bmatrix} := \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{bmatrix}\begin{bmatrix} p_0 \\ \vdots \\ \vdots \\ p_{n-1} \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_n p_n \end{bmatrix}. \tag{18.51}$$

有著名的结论 ([23, 第 124 页]),T_n 有 n 个不同的实特征值 $x_1 < \cdots < x_n$. 考虑特征值 λ 对应的特征向量 $\mathbf{q}$：

$$T_n\mathbf{q} = \lambda\mathbf{q}. \tag{18.52}$$

它等价于 n 个方程

$$\begin{aligned} \alpha_1 q_1 + \beta_1 q_2 &= \lambda q_1, \\ \beta_{k-1}q_{k-1} + \alpha_k q_k + \beta_k q_{k+1} &= \lambda q_k,\ k = 2, \ldots, n-1, \\ \beta_{n-1}q_{n-1} + \alpha_n q_n &= \lambda q_n. \end{aligned} \tag{18.53}$$

因此，向量 $\mathbf{q}$ 的分量满足递推关系式

$$\begin{aligned} q_2 &= \tfrac{\lambda-\alpha_1}{\beta_1} q_1, \\ q_{k+1} &= \tfrac{\lambda-\alpha_k}{\beta_k} q_k - \tfrac{\beta_{k-1}}{\beta_k} q_{k-1},\ k = 2, \ldots, n-1, \end{aligned} \tag{18.54}$$

其中选取 q_1, 使得 $\|\mathbf{q}\|_2 = 1$. 由于 $\beta_k \neq 0$, 显然，每个特征向量的第一个分量是非零的. 否则，整个向量为零，与特征向量的定义矛盾.

另一方面，我们定义

$$\mathbf{p}(x) := \begin{bmatrix} p_0(x) \\ \vdots \\ p_{n-1}(x) \end{bmatrix}. \tag{18.55}$$

从 (18.21) 可知，向量 $\mathbf{p}(\lambda)$ 的分量满足递推关系式

$$\begin{aligned} p_1(\lambda) &= \tfrac{\lambda-\alpha_1}{\beta_1} p_0(\lambda), \\ p_k(\lambda) &= \tfrac{\lambda-\alpha_k}{\beta_k} p_{k-1}(\lambda) - \tfrac{\beta_{k-1}}{\beta_k} p_{k-2}(\lambda),\ k = 2, \ldots, n, \end{aligned} \tag{18.56}$$

除了初始值不同，它与 (18.54) 是相同的. 因而，乘上一个系数，向量 $\mathbf{q}$ 和 $\mathbf{p}(\lambda)$ 一定相等:

$$\mathbf{q} = \frac{q_1}{p_0}\mathbf{p}(\lambda). \tag{18.57}$$

因为 $\beta_n \neq 0$, 所以方程 (18.51) 隐含 $p_n(\lambda) = 0$. 如果对每个特征值 x_k 计算 (18.57), 则可得 (18.50).

18.3 积分法则

考虑构造一个积分法则的问题

$$\int_a^b f(x)\omega(x)\mathrm{d}x \approx \sum_{i=1}^{m+n} w_i f(x_i), \tag{18.58}$$

对不超过最大可能次数的多项式可精确积分. 假设在区间 $[a,b]$ 的边界上，有 m 个横坐标 $x_1 < \cdots < x_m$, 其中 m 只能取 $m = 0, m = 1$ 和 $m = 2$. 令

$$r(x) := \prod_{i=1}^{m}(x - x_i) \tag{18.59}$$

是一个 m 次的多项式，上面 m 个横坐标是它的零点. 而且令

$$s(x) := \prod_{i=m+1}^{m+n}(x - x_i) \tag{18.60}$$

是 n 次的未知多项式，其零点为横坐标 $x_{m+1},\cdots,x_{m+n}$.

下面的定理是计算积分法则的关键 (参见 [20, 第 161 页]).

定理 18.6 *对所有次数不超过 $m+2n-1$ 的多项式 p, Gauss 积分法则 (18.58) 精确成立的充分必要条件是下列两个条件成立:*

1. 对所有次数不超过 $m+n-1$ 的多项式 p, Gauss 积分法则 (18.58) 精确成立.

2. 方程

$$\int_a^b r(x)s(x)p(x)\omega(x)\mathrm{d}x=0 \tag{18.61}$$

可用于所有次数不超过 $n-1$ 的多项式 p.

证明 首先假设，对所有次数不超过 $m+2n-1$ 的多项式， Gauss 积分法则 (18.58) 精确成立. 显然，对所有次数不超过 $m+n-1$ 的多项式， Gauss 积分法则 (18.58) 也精确成立. 多项式 $r(x)s(x)p(x)$ 的次数小于等于 $m+2n-1$, 因此积分精确成立:

$$\int_a^b r(x)s(x)p(x)\omega(x)\mathrm{d}x=\sum_{i=1}^{m+n}w_ir(x_i)s(x_i)p(x_i)=0.$$

另一方面，假设条件 1 和 2 成立. 根据

$$t(x)=r(x)s(x)p(x)+q(x),$$

可分解一个次数小于等于 $m+2n-1$ 的多项式 $t(x)$, 其中 p 和 q 分别是次数小于等于 $n-1$ 和次数小于等于 $m+n-1$ 的多项式. 因而有

$$\begin{aligned}\int_a^b t(x)\omega(x)\mathrm{d}x &= \int_a^b r(x)s(x)p(x)\omega(x)\mathrm{d}x+\int_a^b q(x)\omega(x)\mathrm{d}x\\ &= \int_a^b q(x)\omega(x)\mathrm{d}x.\end{aligned}$$

但是对所有次数小于等于 $m+n-1$ 的多项式，积分法则是精确的. 因为 $t(x_i)=q(x_i)$, 于是有

$$\int_a^b q(x)\omega(x)\mathrm{d}x=\sum_{i=1}^{m+n}w_iq(x_i)=\sum_{i=1}^{m+n}w_it(x_i),$$

从而

$$\int_a^b t(x)\omega(x)\mathrm{d}x=\sum_{i=1}^{m+n}w_it(x_i),$$

对所有次数不超过 $m+2n-1$ 的多项式， Gauss 积分法则精确成立.

现在考虑 $m=0, m=1$ 和 $m=2$ 时，积分法则的构造. 为此，我们必须确定 $m+n$ 个横坐标 x_i, 使得定理 18.6 的条件 2 满足. 对于三种情况，将调用不同的过程.

之后，我们必须确定权 w_i, 使得定理 18.6 的条件 1 满足. 在第 18.7 节中，将解决此问题. MAPLE 算法 18.4, 18.5 和 18.6 将给出这些结果.

18.4 Gauss 积分法则

在 Gauss 积分法则的情况中，没有规定横坐标，即 $m=0$ 和 $r(x)=1$. 如果取 $s(x)=p_n(x)$, 则定理 18.6 的条件 2 满足. 因此， $p_n(x)$ 的零点，即矩阵 T_n 的特征值，表示所求的横坐标 x_i(参见 [18]).

算法 18.4 Gauss 积分法则

```
Gauss := proc (mu0, alpha, beta, n, x, w)
  local D, i, Q, T;

  T := array (1..n, 1..n, symmetric, [[0$n]$n]);
  for i from 1 to n do
    T[i,i] := alpha[i];
  od;
  for i from 1 to n-1 do
    T[i,i+1] := beta[i];
    T[i+1,i] := beta[i];
  od;

  D := evalf (Eigenvals (T, Q));
  x := array (1..n, [seq (D[i], i=1..n)]);
  w := array (1..n, [seq (evalf (mu0) * Q[1,i]^2, i=1..n)]);

  RETURN (NULL);
end:
```

18.5 Gauss-Radau 积分法则

在 Gauss-Radau 积分法则中，在区间 $[a,b]$ 的边界上，仅规定一个横坐标 x_1，即 $m=1$ 和 $r(x)=x-a$ 或 $r(x)=x-b$. 不直接计算满足定理 18.6 的条件 2 的多项式 $s(x)$, 而是构造 $n+1$ 次的多项式 $\tilde{p}_{n+1}(x)=r(x)s(x)$. 从下面两个要求，可确定此多项式：

1. s 是 $\tilde{p}_{n+1}$ 的一个因子：
$$\tilde{p}_{n+1}(x_1)=0. \tag{18.62}$$
2. 方程
$$\int_a^b \tilde{p}_{n+1}(x)p(x)\omega(x)\mathrm{d}x=0 \tag{18.63}$$

可用于所有次数不超过 $n-1$ 的多项式 p.

我们从下面 $\tilde{p}_{n+1}(x)$ 的隐式 (参见 [16]) 开始：

$$x\begin{bmatrix} p_0 \\ \vdots \\ \vdots \\ p_n \end{bmatrix} := \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \alpha_n & \beta_n \\ & & \beta_n & \tilde{\alpha}_{n+1} \end{bmatrix} \begin{bmatrix} p_0 \\ \vdots \\ \vdots \\ p_n \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \tilde{p}_{n+1} \end{bmatrix}, \tag{18.64}$$

其中 $\tilde{\alpha}_{n+1}$ 的值仍需确定.

首先我们说明，对任意的 $\tilde{\alpha}_{n+1}$，选取的 $\tilde{p}_{n+1}$ 满足 (18.63). 由于方程 (18.64)

$$xp_n = \beta_n p_{n-1} + \tilde{\alpha}_{n+1} p_n + \tilde{p}_{n+1}$$

和递推关系式 (18.21)

$$xp_n = \beta_n p_{n-1} + \alpha_{n+1} p_n + \beta_{n+1} p_{n+1},$$

多项式 $\tilde{p}_{n+1}$ 可表示为

$$\tilde{p}_{n+1} = (\alpha_{n+1} - \tilde{\alpha}_{n+1}) p_n + \beta_{n+1} p_{n+1}. \tag{18.65}$$

显然，多项式 $\tilde{p}_{n+1}$ 是 p_n 和 p_{n+1} 的一个线性组合，总是满足 (18.63).

我们只需确定 $\tilde{\alpha}_{n+1}$ 的值，使得 $\tilde{p}_{n+1}(x_1) = 0$，或等价于 x_1 是方程 (18.64) 矩阵的一个特征值. 为此考虑特征值方程

$$x_1 \left[\begin{array}{c} \mathbf{y} \\ \hline \eta \end{array}\right] = \left[\begin{array}{c|c} T_n & \begin{array}{c} \\ \beta_n \end{array} \\ \hline \begin{array}{cc} & \beta_n \end{array} & \tilde{\alpha}_{n+1} \end{array}\right] \left[\begin{array}{c} \mathbf{y} \\ \hline \eta \end{array}\right],$$

或等价于

$$x_1 \mathbf{y} = T_n \mathbf{y} + \beta_n \eta \mathbf{e}_n, \tag{18.66}$$

$$x_1 \eta = \beta_n y_n + \tilde{\alpha}_{n+1} \eta. \tag{18.67}$$

由 (18.66)，可知 $\mathbf{y}$ 是 $(T_n - x_1 I)^{-1} \mathbf{e}_n$ 的倍数. 因此，令

$$\mathbf{y} := (T_n - x_1 I)^{-1} \mathbf{e}_n. \tag{18.68}$$

于是，由 (18.66) 可得

$$\eta = -\frac{1}{\beta_n}. \tag{18.69}$$

注意，依赖于 x_1 的选择，矩阵 $T_n - x_1 I$ 是正定的或负定的. 在每一种情况，向量 $\mathbf{y}$ 的值是唯一确定的.

将 (18.69) 代入 (18.67)，可得

$$\tilde{\alpha}_{n+1} = x_1 + \beta_n^2 y_n. \tag{18.70}$$

因而，我们已经确定多项式 $\tilde{p}_{n+1}(x)$，它的零点是 Gauss-Radau 积分法则的横坐标.

算法 18.5 Gauss-Radau 积分法则

```
Radau := proc (mu0, alpha, beta, n, x1, x, w)
  local alphan1tilde, D, e, i, Q, T, y;

  T := array (1..n, 1..n, symmetric, [[0$n]$n]);
  for i from 1 to n do
    T[i,i] := alpha[i] - x1;
  od;
  for i from 1 to n-1 do
```

```
    T[i,i+1] := beta[i];
    T[i+1,i] := beta[i];
  od;

  e := linalg[vector] (n, 0);  e[n] := 1;
  y := linalg[linsolve] (T, e);

  alphan1tilde := simplify (x1 + beta[n]^2 * y[n]);

  T := array (1..n+1, 1..n+1, symmetric, [[0$n+1]$n+1]);
  for i from 1 to n do
    T[i,i] := alpha[i];
  od;
  T[n+1,n+1] := alphan1tilde;
  for i from 1 to n do
    T[i,i+1] := beta[i];
    T[i+1,i] := beta[i];
  od;

  D := evalf (Eigenvals (T, Q));
  x := array (1..n+1, [seq (D[i], i=1..n+1)]);
  w := array (1..n+1, [seq (evalf (mu0) * Q[1,i]^2, i=1..n+1)]);

  RETURN (NULL);
end:
```

18.6 Gauss-Lobatto 积分法则

在 Gauss-Lobatto 积分法则的情况中，我们规定两个横坐标 $x_1 = a$ 和 $x_2 = b$, 即 $m = 2$ 和 $r(x) = (x-a)(x-b)$. 不直接计算满足定理 18.6 的条件 2 的多项式 $s(x)$, 而是构造 $n+2$ 次的多项式 $\tilde{p}_{n+2}(x) = r(x)s(x)$. 从下面两个要求，可确定此多项式：

1. s 是 $\tilde{p}_{n+2}$ 的一个因子：

$$\tilde{p}_{n+2}(x_1) = \tilde{p}_{n+2}(x_2) = 0. \tag{18.71}$$

2. 方程

$$\int_a^b \tilde{p}_{n+2}(x)p(x)\omega(x)\mathrm{d}x = 0 \tag{18.72}$$

可用于所有次数不超过 $n-1$ 的多项式 p.

我们选取多项式 $\tilde{p}_{n+2}$, 它由矩阵方程隐式地定义 (参见 [16]):

$$x\begin{bmatrix} p_0 \\ \vdots \\ \vdots \\ p_n \\ \tilde{p}_{n+1} \end{bmatrix} := \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \beta_n & \\ & & \beta_n & \alpha_{n+1} & \tilde{\beta}_{n+1} \\ & & & \tilde{\beta}_{n+1} & \tilde{\alpha}_{n+2} \end{bmatrix}\begin{bmatrix} p_0 \\ \vdots \\ \vdots \\ p_n \\ \tilde{p}_{n+1} \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ \tilde{p}_{n+2} \end{bmatrix}, \tag{18.73}$$

其中 $\tilde{\beta}_{n+1} \neq 0$, $\tilde{\alpha}_{n+2}$ 仍没有确定.

现在我们说明, 对任意的 $\tilde{\beta}_{n+1}$ 和 $\tilde{\alpha}_{n+2}$, 选取的 $\tilde{p}_{n+2}$ 满足 (18.72). 根据 (18.21), (18.73) 的倒数第二个方程

$$xp_n = \beta_n p_{n-1} + \alpha_{n+1}p_n + \tilde{\beta}_{n+1}\tilde{p}_{n+1}$$

定义多项式 $\tilde{p}_{n+1}$ 是 p_{n+1} 的倍数:

$$\tilde{p}_{n+1} = \frac{\beta_{n+1}}{\tilde{\beta}_{n+1}}p_{n+1}. \tag{18.74}$$

将 (18.74) 代入 (18.73) 的最后一个方程

$$x\tilde{p}_{n+1} = \tilde{\beta}_{n+1}p_n + \tilde{\alpha}_{n+2}\tilde{p}_{n+1} + \tilde{p}_{n+2},$$

可得

$$x\frac{\beta_{n+1}}{\tilde{\beta}_{n+1}}p_{n+1} = \tilde{\beta}_{n+1}p_n + \tilde{\alpha}_{n+2}\frac{\beta_{n+1}}{\tilde{\beta}_{n+1}}p_{n+1} + \tilde{p}_{n+2}.$$

由于

$$xp_{n+1} = \beta_{n+1}p_n + \alpha_{n+2}p_{n+1} + \beta_{n+2}p_{n+2},$$

因此可得多项式

$$\tilde{p}_{n+2} = (\frac{\beta_{n+1}^2}{\tilde{\beta}_{n+1}} - \tilde{\beta}_{n+1})p_n + \frac{\beta_{n+1}}{\tilde{\beta}_{n+1}}(\alpha_{n+2} - \tilde{\alpha}_{n+2})p_{n+1} + \frac{\beta_{n+1}\beta_{n+2}}{\tilde{\beta}_{n+1}}p_{n+2}. \tag{18.75}$$

显然, 多项式 $\tilde{p}_{n+2}$ 是 p_n, p_{n+1} 和 p_{n+2} 的一个线性组合, 总是满足方程 (18.72).

现在, 我们将确定 $\tilde{\beta}_{n+1} \neq 0$ 和 $\tilde{\alpha}_{n+2}$ 的值, 使得 $\tilde{p}_{n+2}(a) = \tilde{p}_{n+2}(b) = 0$. 但是这等价于, 要求方程 (18.73) 的矩阵有两个特征值 a 和 b. 考虑特征值方程

$$a\left[\begin{array}{c} \mathbf{y} \\ \hline \eta \end{array}\right] = \left[\begin{array}{c|c} T_{n+1} & \begin{array}{c} \\ \tilde{\beta}_{n+1} \end{array} \\ \hline \begin{array}{cc} & \tilde{\beta}_{n+1} \end{array} & \tilde{\alpha}_{n+2} \end{array}\right]\left[\begin{array}{c} \mathbf{y} \\ \hline \eta \end{array}\right]$$

和

$$b\left[\begin{array}{c} \mathbf{z} \\ \hline \zeta \end{array}\right] = \left[\begin{array}{c|c} T_{n+1} & \begin{array}{c} \\ \tilde{\beta}_{n+1} \end{array} \\ \hline \begin{array}{cc} & \tilde{\beta}_{n+1} \end{array} & \tilde{\alpha}_{n+2} \end{array}\right]\left[\begin{array}{c} \mathbf{z} \\ \hline \zeta \end{array}\right],$$

或等价于

$$
\begin{aligned}
a\mathbf{y} &= T_{n+1}\mathbf{y} + \tilde{\beta}_{n+1}\eta\mathbf{e}_{n+1}, &(18.76)\\
a\eta &= \tilde{\beta}_{n+1}y_{n+1} + \tilde{\alpha}_{n+2}\eta, &(18.77)\\
b\mathbf{z} &= T_{n+1}\mathbf{z} + \tilde{\beta}_{n+1}\zeta\mathbf{e}_{n+1}, &(18.78)\\
b\zeta &= \tilde{\beta}_{n+1}z_{n+1} + \tilde{\alpha}_{n+2}\zeta. &(18.79)
\end{aligned}
$$

由方程 (18.76) 和 (18.78), 可知向量 $\mathbf{y}$ 和 $\mathbf{z}$ 分别是 $(T_{n+1}-aI)^{-1}\mathbf{e}_{n+1}$ 和 $(T_{n+1}-bI)^{-1}\mathbf{e}_{n+1}$ 的倍数. 如果定义 $\mathbf{y}$ 和 $\mathbf{z}$

$$
\begin{aligned}
\mathbf{y} &:= (T_{n+1}-aI)^{-1}\mathbf{e}_{n+1}, &(18.80)\\
\mathbf{z} &:= (T_{n+1}-bI)^{-1}\mathbf{e}_{n+1}, &(18.81)
\end{aligned}
$$

则方程 (18.76) 和 (18.78) 隐含

$$\eta = \zeta = -\frac{1}{\tilde{\beta}_{n+1}}. \tag{18.82}$$

而且矩阵 $T_{n+1}-aI$ 和 $T_{n+1}-bI$ 分别是正定和负定矩阵. 特别地, 我们有不等式

$$y_{n+1} > 0 > z_{n+1}. \tag{18.83}$$

将 (18.82) 代入 (18.77) 和 (18.79), 再用 $\tilde{\beta}_{n+1}$ 相乘, 可得线性系统

$$\begin{bmatrix} 1 & -y_{n+1} \\ 1 & -z_{n+1} \end{bmatrix} \begin{bmatrix} \tilde{\alpha}_{n+2} \\ \tilde{\beta}_{n+1}^2 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}. \tag{18.84}$$

因为 (18.83), 所以线性系统 (18.84) 有唯一解

$$
\begin{aligned}
\tilde{\alpha}_{n+2} &= \frac{by_{n+1} - az_{n+1}}{y_{n+1} - z_{n+1}}, &(18.85)\\
\tilde{\beta}_{n+1} &= \sqrt{\frac{b-a}{y_{n+1} - z_{n+1}}} > 0. &(18.86)
\end{aligned}
$$

因而, 最终确定多项式 $\tilde{p}_{n+2}(x)$, 它的零点是 Gauss-Lobatto 积分法则的横坐标.

算法 18.6 Gauss-Lobatto 积分法则

```
Lobatto := proc (mu0, alpha, beta, n, x1, x2, x, w)
  local alphan2tilde, betan1tilde, D, e, i, Q, T, y, z;

  T := array (1..n+1, 1..n+1, symmetric, [[0$n+1]$n+1]);
  for i from 1 to n+1 do
    T[i,i] := alpha[i] - x1;
  od;
  for i from 1 to n do
    T[i,i+1] := beta[i];
    T[i+1,i] := beta[i];
```

```
    od;

    e := linalg[vector] (n+1, 0);  e[n+1] := 1;
    y := linalg[linsolve] (T, e);

    for i from 1 to n+1 do
      T[i,i] := alpha[i] - x2;
    od;
    z := linalg[linsolve] (T, e);
    alphan2tilde := simplify ((x2 * y[n+1] - x1*z[n+1]) /
                              (y[n+1] - z[n+1]));
    betan1tilde := simplify (sqrt ((x2-x1) /
                              (y[n+1] - z[n+1])));

    T := array (1..n+2, 1..n+2, symmetric, [[0$n+2]$n+2]);
    for i from 1 to n+1 do
      T[i,i] := alpha[i];
    od;
    T[n+2,n+2] := alphan2tilde;
    for i from 1 to n do
      T[i,i+1] := beta[i];
      T[i+1,i] := beta[i];
    od;
    T[n+1,n+2] := betan1tilde;
    T[n+2,n+1] := betan1tilde;
    D := evalf (Eigenvals (T, Q));
    x := array (1..n+2, [seq (D[i], i=1..n+2)]);
    w := array (1..n+2, [seq (evalf (mu0) * Q[1,i]^2, i=1..n+2)]);
    RETURN (NULL);
  end:
```

18.7 权

计算权的过程与上面所述每个积分法则的过程相同. 我们必须确定权 $w_1, \ldots, w_{m+n}$, 使得定理 18.6 的条件 1 成立. 等价地, 我们要求 (18.64) 和 (18.73) 中的多项式 $p_0, \ldots, p_n$ 和 $\tilde{p}_{n+1}, \ldots, \tilde{p}_{m+n-1}$ 可精确地积分:

$$
\begin{aligned}
\int_a^b p_k(x)\omega(x)\mathrm{d}x &= \textstyle\sum_{i=1}^{m+n} w_i p_k(x_i) = \frac{1}{p_0}\delta_{k0}, \quad k = 0, \ldots, \min(n, m+n-1), \\
\int_a^b \tilde{p}_k(x)\omega(x)\mathrm{d}x &= \textstyle\sum_{i=1}^{m+n} w_i \tilde{p}_k(x_i) = 0, \qquad k = n+1, \ldots, m+n-1.
\end{aligned}
$$

用矩阵的形式可写成

$$\begin{bmatrix} p_0(x_1) & \cdots & \cdots & p_0(x_{m+n}) \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ \tilde{p}_{m+n-1}(x_1) & \cdots & \cdots & \tilde{p}_{m+n-1}(x_{m+n}) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ \vdots \\ w_{m+n} \end{bmatrix} = \begin{bmatrix} 1/p_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

或 $P\mathbf{w} = \mathbf{e}_1/p_0$. 因为定理 18.5, 可得 $Q = PW$, 使得

$$\mathbf{w} = \frac{1}{p_0} W Q^{\mathrm{T}} \mathbf{e}_1.$$

由 (18.14) 和 (18.6) 可得

$$\int_a^b p_0^2(x)\omega(x)\mathrm{d}x = p_0^2\mu_0 = 1,$$

最后得到权 w_i

$$w_i = \mu_0 q_{1i}^2, \quad i = 1, \ldots, m+n. \tag{18.87}$$

现在作为一个应用，我们计算关于区间 $[1, 2]$ 和权函数

$$\omega(x) := \frac{1}{x} \tag{18.88}$$

的 Gauss-Lobatto 积分法则.

下面的 MAPLE 语句，计算正交多项式的递推关系式 (18.21). 然后，调用算法 18.6 即可计算积分法则:

```
> N := 11:
> Digits := 100:
> a := 1:
> b := 2:
> omega := x -> 1 / x:

> alphahat := array (1..2*N-1, [0$2*N-1]):
> betahat := array (1..2*N-1, [1$2*N-1]):
> gammahat := array (1..2*N-2, [0$2*N-2]):

> pi0 := 1:
> nu[0] := ln (b) - ln (a):
> for n from 1 to 2*N-1 do
>   nu[n] := (b^n - a^n) / n;
> od:
> mu0 := nu[0] / pi0:

> SackDonovan (pi0, alphahat, betahat, gammahat, nu, N,
>              'alpha', 'beta');
```

```
> Lobatto (mu0,evalf(op(alpha)),evalf(op(beta)),N-1,a,b,
>           'x', 'w');
```

横坐标和权如表 18.3 所示，结果保留 20 位十进制数字.

表 18.3 Gauss-Lobatto 积分法则的横坐标和权.

k	x_k	w_k
1	1.0000000000000000000	0.0073101161434772987648
2	1.0266209022507358746	0.043205676918139196040
3	1.0875753284733094064	0.070685480638318232547
4	1.1786507622317286926	0.088433599898292821869
5	1.2936151328648399126	0.096108531378420052625
6	1.4243072396520977438	0.095097966102219702354
7	1.5611522948208446896	0.087445652257865527102
8	1.6938824042052787649	0.075118353999491464492
9	1.8123676368833335432	0.059695944656549921510
10	1.9074586992416619710	0.042332113537812345903
11	1.9717495114038218100	0.023827527632972240207
12	2.0000000000000000000	0.0038862173963865060038

18.8 积分误差

现在，我们考虑怎样表示积分误差

$$E[f] := \int_a^b f(x)\omega(x)\mathrm{d}x - \sum_{i=1}^{m+n} w_i f(x_i). \tag{18.89}$$

假设在区间 $[a,b]$ 上，被积函数 f 是一个 $(m+2n)$ 次连续可微函数. 而且令 $H(x)$ 是唯一确定的次数小于等于 $m+2n-1$ 的多项式，它满足 Hermite 插值条件 (参见 [29, 第 44 页])

$$\begin{aligned} H(x_i) &= f(x_i), \quad i=1,\ldots,m+n, \\ H'(x_i) &= f'(x_i), \quad i=m+1,\ldots,m+n. \end{aligned}$$

首先我们检查，在区间 $[a,b]$ 上，多项式 $H(x)$ 近似给定函数 $f(x)$ 的情况.

定理 18.7 令

$$Q(x) := \prod_{i=1}^{m}(x-x_i) \cdot \prod_{i=m+1}^{m+n}(x-x_i)^2 \tag{18.90}$$

是一个 $m+2n$ 次的多项式，则对于区间 $[a,b]$ 内的任意 x，存在 $\xi \in (a,b)$，使得

$$f(x) = H(x) + \frac{f^{(m+2n)}(\xi)}{(m+2n)!}Q(x). \tag{18.91}$$

证明 对于 $x=x_i$，方程 (18.91) 显然成立. 因此我们假设 $x \neq x_i$ 和 $x \in [a,b]$. 考虑函数

$$F(t) = f(t) - H(t) - \frac{f(x)-H(x)}{Q(x)}Q(t). \tag{18.92}$$

显然，在区间 $[a,b]$ 上，$F(t)$ 至少有 $m+2n+1$ 个零点. 由 Rolle 定理可知，在区间 $[a,b]$ 的内部，$F^{(m+2n)}(t)$ 至少有一个零点 ξ：

$$F^{(m+2n)}(\xi) = f^{(m+2n)}(\xi) - \frac{f(x)-H(x)}{Q(x)}(m+2n)! = 0.$$

于是方程 (18.91) 成立.

类似的证明，读者可参见 [20, 第 49 页] 和 [29, 第 48 页].

有了这些准备之后，我们可给出积分误差 (18.89) 的表达式. 参见 [20, 第 162-163 页] 和 [29, 第 134 页].

定理 18.8 *积分法则 (18.58) 的积分误差是*

$$E[f] = \frac{f^{(m+2n)}(\xi)}{(m+2n)!}\int_a^b Q(x)\omega(x)\mathrm{d}x \tag{18.93}$$

其中 $a<\xi<b$.

证明 根据 (18.91), 对于 $a \le x \le b$, 被积函数 $f(x)$ 可表示为

$$f(x) = H(x) + \frac{f^{(m+2n)}(\xi(x))}{(m+2n)!}Q(x) \tag{18.94}$$

其中 $a<\xi(x)<b$. 由于

$$f^{(m+2n)}(\xi(x)) = (m+2n)!\frac{f(x)-H(x)}{Q(x)},$$

对于 $a \le x \le b$, 我们有一个连续函数 $f^{(m+2n)}(\xi(x))$.

对 $f(x)$ 的表达式 (18.94) 积分，可得到

$$\begin{aligned}\int_a^b f(x)\omega(x)\mathrm{d}x &= \textstyle\int_a^b H(x)\omega(x)\mathrm{d}x \\ &\quad + \textstyle\frac{1}{(m+2n)!}\int_a^b f^{(m+2n)}(\xi(x))Q(x)\omega(x)\mathrm{d}x.\end{aligned} \tag{18.95}$$

插值多项式 $H(x)$ 的次数小于等于 $m+2n-1$, 因此它被精确地积分：

$$\int_a^b H(x)\omega(x)\mathrm{d}x = \sum_{i=1}^{m+n} w_i H(x_i) = \sum_{i=1}^{m+n} w_i f(x_i). \tag{18.96}$$

另一方面，在区间 $[a,b]$ 上，$Q(x)$ 的符号是固定的. 利用广义的积分中值定理 [19, 第 477 页], 对于 $a<\xi<b$, 可得

$$\int_a^b f^{(m+2n)}(\xi(x))Q(x)\omega(x)\mathrm{d}x = f^{(m+2n)}(\xi)\int_a^b Q(x)\omega(x)\mathrm{d}x. \tag{18.97}$$

将 (18.96) 和 (18.97) 代入 (18.95), 得到

$$\int_a^b f^{(m+2n)}(\xi(x))\mathrm{d}x = \sum_{i=1}^{m+n} w_i f(x_i) + \frac{f^{(m+2n)}(\xi)}{(m+2n)!}\int_a^b Q(x)\omega(x)\mathrm{d}x \tag{18.98}$$

其中 $a<\xi<b$. 于是方程 (18.93) 成立.

参考文献

[1] N.I. AKHIEZER, *The Classical Moment Problem*, translated by N. Kemmer, Oliver & Boyd, Edinburgh and London, 1965.

[2] J. BOUZITAT, *Sur l'intégration numérique approchée par la méthode de Gauss généralisée et sur une extension de cette méthode*, C R Acad Sci Paris, 229(1949), pp. 1201-1203.

[3] E.B. CHRISTOFFEL, *Über die Gaussische Quadratur und eine Verallgemeinerung derselben*, J. Reine Angew. Math., 55(1958), pp. 61-82.

[4] P.J. DAVIS AND P. RABINOWITZ,*Methods of Numerical Integration*, Academic Press, Orlando, 1984.

[5] R.N. DESMARAIS, *Programs for computing abscissas and weights for classical and non-classical Gaussian quadrature formulas*, NASA Report TN D-7924, NASA Langley Research Center, Hampton VA, 1975.

[6] A. ERDÉLYI ET AL., *Higher Transcendental Functions*, Bateman Manuscript Project, McGraw-Hill, New York, 1953.

[7] C.F. GAUSS, *Methods Nova Integralium Valores per Approximationem Inveniendi*, Werke, Vol. 3, Göttingen, 1866, pp. 163-196.

[8] W. GAUTSCHI, *Computational aspects of three-term recurrence relations*, SIAM Review, 9(1967), pp. 24-82.

[9] W. GAUTSCHI, *Construction of Gauss-Christoffel Quadrature Formulas*, Math. Comp., 22(1968), pp. 251-270.

[10] W. GAUTSCHI, *On the Construction of Gaussian Quadrature Rules from Modified Moments*, Math. Comp., 24(1970), pp. 245-260.

[11] W. GAUTSCHI, *Minimal Solutions of Three-Term Recurrence Relations and Orthogonal Polynomials*, Math. Comp., 36(1981), pp. 547-554.

[12] W. GAUTSCHI, *A Survey of Gauss-Christoffel Quadrature Formulae*, in E. B. Christoffel, The Influence of His Work on Mathematics and the Physical Sciences, ed. P. L. Butzer and F. Fehér, Birkhäuser, Basel, 1981, pp. 72-147.

[13] W. GAUTSCHI, *An algorithmic implementation of the generalized Christoffel theorem*, in Numerical Integration, Internat. Ser. Numer. Math., ed. G. Hämmerlin, Birkhäuser, Basel, 57(1982), pp. 89-106.

[14] W. GAUTSCHI, *Questions of numerical condition related to polynomials*, in Studies in Mathematics, Vol. 24: Studies in Numerical Analysis, ed. G. H. Golub, Math. Assoc. Amer., Washington, 1984, pp. 140-177.

[15] W. GAUTSCHI, *Orthogonal polynomials-constructive theory and applications*, J. Comput. Appl. Math., 12&13(1985), pp. 61-76.

[16] G. H. GOLUB, *Some Modified Matrix Eigenvalue Problems*, SIAM Review, 15(1973), pp. 318-334.

[17] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Second Edition, The Johns Hopkins University Press, Baltimore, 1989.

[18] G. H. GOLUB AND J. H. WELSCH, *Calculation of Gauss Quadrature Rules*, Math. Comp., 23(1969), pp. 221-230.

[19] H. HEUSER, *Lehrbuch der Analysis, Teil 1*, Teubner, Stuttgart, 1986.

[20] V. I. KRYLOV, *Approximate Calculation of Integrals*, translated by A. H. Stroud, Macmillan, New York, 1962.

[21] A. MARKOFF, *Sur la méthode de Gauss pour le calcul approché des intégrales*, Math. Ann., 25(1885), pp. 427-432.

[22] I. P. MYSOVSKIH, *On the Construction of Cubature Formulas with Fewest Nodes*, Soviet Math. Dokl., 9(1968), pp. 277-280.

[23] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, 1980.

[24] W. H. PRESS AND S. A. TEUKOLSKY, *Orthogonal Polynomials and Gaussian Quadrature with Non-classical Weight Functions*, Computers in Physics, 4(1990), pp. 423-426.

[25] P. RABINOWITZ, *Abscissas and Weights for Lobatto Quadrature of High Order*, Math. Comp., 14(1960), pp. 47-52.

[26] R. RADAU, *Étude sur les formules d'approximation qui servent à calculer la valeur numérique d'une intégrale définie*, J. Math. Pures. Appl., Ser. 3, 6(1880), pp. 283-336.

[27] R. A. SACK AND A. F. DONOVAN, *An Algorithm for Gaussian Quadrature given Modified Moments*, Numer. Math., 18(1972), pp. 465-478.

[28] J. A. SHOHAT AND J. D. TAMARKIN, *The Problem of Moments*, Second Edition, American Mathematical Society, New York, 1950.

[29] J. STOER, *Einführung in die Numerische Mathmetik I*, Springer-Verlay, 1983.

[30] J. STOER AND R. BULIRSCH, *Einführung in die Numerische Mathmetik II*, Springer-Verlay, 1978.

[31] A. H. STROUD AND D. SECREST, *Gaussian Quadrature Formulars*, Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

[32] G. SZEGŐ, *Orthogonal Polynomials*, American Mathematical Society, New York, 1939.

[33] H. S. WALL, *Analytic Theory of Continued Fractions*, van Nostrand, New York, 1948.

[34] J. C. WHEELER, *Modified moments and Gaussian quadratures*, Rocky Mountain J. of Math., 4(1974), pp. 287-296.

第十九章　Runge-Kutta 显式公式的符号计算

D. Gruntz

19.1　引言

在本章中，我们说明怎样利用 MAPLE 得到 Runge-Kutta 显式公式. 在数值分析中，它用于求解一阶微分方程组. 我们将说明怎样构造 Runge-Kutta 显式公式系数的非线性方程组，以及怎样求解它. 在本章末尾，我们给出构造给定大小和阶数的 Runge-Kutta 显式公式的全部过程. 我们将看到，这样用途广泛的程序能求解多大的方程.

用 x_k 附近的 Taylor 级数，可以近似初始值问题

$$y'(x) = f(x, y(x)), \quad y(x_k) = y_k \tag{19.1}$$

的解. 即通过重复微分法，并在每次 $y'(x)$ 出现时都用 $f(x, y(x))$ 代替.

$$\begin{aligned} y(x_k+h) &= \sum_{i=0}^{\infty} y^{(i)}(x_k)\tfrac{h^i}{i!} \\ &= y(x_k) + hf(x_k, y(x_k)) + \tfrac{h^2}{2}(\tfrac{\partial}{\partial x} f(x, y(x))|_{x=x_k}) + \ldots \\ &= y(x_k) + h\underbrace{(f(x_k, y(x_k)) + \frac{h}{2}(f_x(x_k, y(x_k)) + f(x_k, y(x_k))f_y(x_k, y(x_k))) + \ldots)}_{\Phi(x_k, y(x_k), h)} \end{aligned} \tag{19.2}$$

只利用 (19.2) 中的几项，其中 h 是充分小，并从点 $x_{k+1} = x_k + h$ 重复计算，便可计算出 (19.1) 解的数值近似.

Runge-Kutta 方法的思想是，只利用 $f(x, y(x))$ 的值，而不用它的导数，来近似阶数不超过 m 的 Taylor 级数 (19.2). 例如，改进 Euler 方法的 Taylor 级数，

$$y_{k+1} = y_k + hf(x_k + \frac{h}{2}, y_k + \frac{h}{2} f(x_k, y_k)), \tag{19.3}$$

与 (19.2) 中不超过 h^2 的系数均相同，即直到 $m = 2$ 阶. 在本章后面，将证明此结论.

Kutta[9] 用公式表示出 Runge-Kutta 公式的一般方案，有如下形式：假设 s 是一个整数 (“阶段数”),$a_{i,j}, b_i, c_i$ 是实系数. 于是称方法

$$\begin{aligned} k_1 &= f(x, y), \\ k_2 &= f(x + c_2 h, y + h a_{2,1} k_1), \\ &\vdots \\ k_s &= f(x + c_s h, y + h \sum_{j=1}^{s-1} a_{s,j} k_j), \\ \Phi(x, y, h) &= \sum_{i=1}^{s} b_i k_i, \\ y_{k+1} &= y_k + h\Phi(x_k, y_k, h), \end{aligned}$$

是一个 s 阶段的 Runge-Kutta 显式方法. s 也是用于计算 y_{k+1} 所需 f 值的个数. 这样一个 Runge-Kutta 方案, 可用 $(s^2+3s-2)/2$ 个 $a_{i,j}, b_i, c_i$ 变量定义. 在文献中, 用下列系数表表示这样的方法, 已经成为惯例:

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{2,1} & & & & \\
c_3 & a_{3,1} & a_{3,2} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}
$$

一个 m 阶的 s 阶段 Runge-Kutta 方法是由下面两步组成:

1. 由条件, Runge-Kutta 公式和解的 Taylor 级数必须在 m 阶以下要一致, 以便能构造出参数为 $a_{i,j}, b_i, c_i$ 的非线性方程组, 而且
2. 此方程组要么可解, 要么证明是不相容的.

到目前为止, 第二步更加困难 (参见 [12, 4, 2]). 在 SIGSAM Bulletin[8] 的问题部分中, 提出了产生 Runge-Kutta 方程的问题, 但是从没有答案.

在下一节, 对于 $s=3$ 和 $m=3$, 我们将用 MAPLE 解第一步和第二步.

19.2 参数方程的导出

为了得到 Runge-Kutta 公式中参数的方程组, 我们必须计算解 $y(x+h)$ 和 Runge-Kutta 公式的 Taylor 级数. MAPLE 知道怎样对两个参数都依赖 x 的函数求微分, 即

```
> diff(f(x, y(x)), x);
```

$$D_1(f)(x, \mathrm{y}(x)) + D_2(f)(x, \mathrm{y}(x))\,(\frac{\partial}{\partial x}\,\mathrm{y}(x))$$

其中 $D_1(f)(x, y(x))$ 是 f 关于第一项的微分, 即 $f_x(x, y(x))$. 对 MAPLE 中 D 算子的进一步讨论, 读者参见 [10, 11].

下一个问题是, 当 $y'(x)$ 出现时, 用 $f(x, y(x))$ 代替它. 有几种处理方法. 一种方法是利用过程 diff 提供的界面, 它允许用户安装自己的微分函数. (注意: 因为 diff 记得每个函数调用, 所以只有重新启动 MAPLE, 才能生效.)

```
> restart;
> `diff/y` := (a, x) -> f(a, y(a))*diff(a, x):
> diff(y(x), x);
```

$$\mathrm{f}(x, \mathrm{y}(x))$$

```
> diff(y(x), x$2);
```

$$D_1(f)(x, \mathrm{y}(x)) + D_2(f)(x, \mathrm{y}(x))\,\mathrm{f}(x, \mathrm{y}(x))$$

另一种方法是用自己的定义，重写算子 y 的微分．当带有参数 y 的 D 被调用时，返回用户定义的过程．用此定义，可计算 $y(x+h)$ 在 $h=0$ 附近的 Taylor 级数，这正是我们所要的．我们认为这是最好的解法：

```
> D(y) := x -> f(x, y(x)):
> taylor(y(x+h), h=0, 3);
```

$$\mathrm{y}(x)+\mathrm{f}(x,\,\mathrm{y}(x))\,h+(\frac{1}{2}\,D_1(f)(x,\,\mathrm{y}(x))+\frac{1}{2}\,D_2(f)(x,\,\mathrm{y}(x))\,\mathrm{f}(x,\,\mathrm{y}(x)))\,h^2+\mathrm{O}(h^3)$$

此级数是正确的，但是，由于 $f(x,y(x))$ 微分的复杂符号，它非常难以阅读．因此我们引入微分的 `alias` 定义，它使得表达式具有可读性．这样我们便得出 $y(x)$ 前几项微分的著名形式：

```
> alias(F = f(x,y(x)),
>    F[x] = D[1](f)(x,y(x)), F[y] = D[2](f)(x,y(x)),
>    F[x,x] = D[1,1](f)(x,y(x)), F[x,y] = D[1,2](f)(x,y(x)),
>    F[y,y] = D[2,2](f)(x,y(x))):
> diff(F, x);
```

$$F_x+F_y\,F$$

```
> diff(F, x$2);
```

$$F_{x,\,x}+F_{x,\,y}\,F+(F_{x,\,y}+F_{y,\,y}\,F)\,F+F_y\,(F_x+F_y\,F)$$

```
> taylor(y(x+h), h=0, 3);
```

$$\mathrm{y}(x)+F\,h+(\frac{1}{2}\,F_x+\frac{1}{2}\,F_y\,F)\,h^2+\mathrm{O}(h^3)$$

为了计算函数 Φ 的 Taylor 级数，它在一般 Runge-Kutta 方案中被定义，我们必须计算表达式 $k_i=f(x+c_ih,y+h\sum_{j=1}^{i-1}a_{i,j}k_j)$ 在 $h=0$ 附近的级数．这也可简单地用命令 `taylor` 执行．对于多变量的 Taylor 级数展开，可用函数 `mtaylor`，但是我们的情况不需要这样做．

```
> taylor(f(x+h, y(x)+h), h=0, 3);
```

$$F+(F_x+F_y)\,h+(\frac{1}{2}\,F_{x,\,x}+F_{x,\,y}+\frac{1}{2}\,F_{y,\,y})\,h^2+\mathrm{O}(h^3)$$

为计算解 $y(x)$ 和 Runge-Kutta 公式的 Taylor 级数，现在我们准备一些工具．然后我们计算 3 阶 $(m=3)$ 3 阶段 $(s=3)$Runge-Kutta 公式的系数．

```
> m := 3:
> taylor(y(x+h), h=0, m+1);
```

$$\mathrm{y}(x)+F\,h+(\frac{1}{2}\,F_x+\frac{1}{2}\,F_y\,F)\,h^2+(\frac{1}{6}\,F_{x,\,x}+\frac{1}{3}\,F_{x,\,y}\,F+\frac{1}{6}\,F_{y,\,y}\,F^2+\frac{1}{6}\,F_y\,F_x+\frac{1}{6}\,{F_y}^2\,F)\,h^3+\mathrm{O}(h^4)$$

```
> TaylorPhi := normal((convert(", polynom) - y(x))/h);
```

$$TaylorPhi:=F+\frac{1}{2}\,h\,F_x+\frac{1}{2}\,h\,F_y\,F+\frac{1}{6}\,h^2\,F_{x,\,x}+\frac{1}{3}\,h^2\,F_{x,\,y}\,F+\frac{1}{6}\,h^2\,F_{y,\,y}\,F^2+\frac{1}{6}\,h^2\,F_y\,F_x+\frac{1}{6}\,h^2\,{F_y}^2\,F$$

convert 命令将 Taylor 级数转变成一个多项式，即它舍去 O 项. 变量 *TaylorPhi* 对应方程 (19.2) 的 Φ.

对于 Runge-Kutta 方案，可得下列 Taylor 级数. 注意，我们保留参数 $a_{i,j}, b_i, c_i$ 仍为符号对象.

```
> k[1]:=taylor(f(x, y(x)), h=0,m):
> k[2]:=taylor(f(x+c[2]*h,y(x)+h*(a[2,1]*k[1])), h=0,m):
> k[3]:=taylor(f(x+c[3]*h,y(x)+h*(a[3,1]*k[1]+a[3,2]*k[2])),h=0,m):
> RungeKuttaPhi := series(b[1]*k[1]+b[2]*k[2]+b[3]*k[3], h, m):
> RungeKuttaPhi := convert(RungeKuttaPhi, polynom);
```

$$
\begin{aligned}
&RungeKuttaPhi := b_1\,F + b_2\,F + b_3\,F \\
&\quad + (b_2\,(F_x\,c_2 + F_y\,a_{2,1}\,F) + b_3\,(F_x\,c_3 + F_y\,a_{3,1}\,F + F_y\,a_{3,2}\,F))\,h + (\\
&\quad b_2\,(\frac{1}{2}\,F_{x,x}\,{c_2}^2 + c_2\,F_{x,y}\,a_{2,1}\,F + \frac{1}{2}\,{a_{2,1}}^2\,F^2\,F_{y,y}) + b_3(\frac{1}{2}\,F_{x,x}\,{c_3}^2 + c_3\,F_{x,y}\,a_{3,1}\,F \\
&\quad + c_3\,F_{x,y}\,a_{3,2}\,F + \frac{1}{2}\,{a_{3,1}}^2\,F^2\,F_{y,y} + a_{3,1}\,F^2\,F_{y,y}\,a_{3,2} + \frac{1}{2}\,{a_{3,2}}^2\,F^2\,F_{y,y} + F_y\,a_{3,2}\,F_x\,c_2 \\
&\quad + {F_y}^2\,a_{3,2}\,a_{2,1}\,F))h^2
\end{aligned}
$$

多项式 *TaylorPhi* 与 *RungeKuttaPhi* 的差 d 应为 0. 考虑 d 是 h, F, F_x, F_y, F_{xx} 等未知数的一个多项式，我们令此多项式的系数为 0. 于是我们得到的非线性方程组 (19.4) 一定可解.

```
> d := expand(TaylorPhi - RungeKuttaPhi):
> eqns := {coeffs(d, [h,F,F[x],F[y],F[x,x],F[x,y],F[y,y]])};
```

$$
\begin{aligned}
eqns\ := \{&\tfrac{1}{2} - b_3c_3 - b_2c_2, -\tfrac{1}{2}b_2{c_2}^2 - \tfrac{1}{2}b_3{c_3}^2 + \tfrac{1}{6}, \\
&\tfrac{1}{6} - b_3a_{3,2}a_{2,1}, -b_3a_{3,2}c_2 + \tfrac{1}{6}, \\
&-b_3a_{3,1}a_{3,2} - \tfrac{1}{2}b_3{a_{3,2}}^2 - \tfrac{1}{2}b_2{a_{2,1}}^2 + \tfrac{1}{6} - \tfrac{1}{2}b_3{a_{3,1}}^2, \\
&1 - b_3 - b_2 - b_1, \tfrac{1}{3} - b_3c_3a_{3,1} - b_2c_2a_{2,1} - b_3c_3a_{3,2}, \\
&-b_3a_{3,2} - b_2a_{2,1} + \tfrac{1}{2} - b_3a_{3,1}\}
\end{aligned}
\tag{19.4}
$$

```
> vars := indets(eqns);
```

$$vars := \{a_{2,1},\, c_2,\, a_{3,1},\, a_{3,2},\, c_3,\, b_1,\, b_2,\, b_3\}$$

19.3 解方程组

在这一节，我们讨论求 Runge-Kutta 公式的第二步，即如何求解非线性方程组 (19.4) 的问题. 注意，我们必须处理一个未知数的多项式方程组. 对于此类问题，有特殊的算法.

我们描述两个算法，它们用计算机代数系统求解多项式方程组. 第一个算法是基于多项式理想的 Gröbner 基理论，第二个算法是利用多项式结式执行非线性消去法. 对这两种方法的进一步介绍参见 [5].

但是，我们首先试试 MAPLE 的 solve 命令. 此命令利用代入法这种自然的方法，求解方程组. 对于 (19.4)，这个方法相当好. 因为后面我们将看到，方程组几乎是三角形形式. 在 [6] 中详细地描述了这种用 MAPLE 的 solve 命令的代入算法.

```
> sols := solve(eqns, vars);
```

$$
\begin{aligned}
sols := \{&b_3 = b_3,\ a_{3,2} = \frac{1}{4}\,\frac{2\,b_3\,c_3 - 1}{b_3\,\%1},\ c_3 = c_3,\ a_{3,1} = \frac{1}{4}\,\frac{-6\,b_3\,c_3 + 1 + 12\,b_3{}^2\,c_3{}^3}{b_3\,\%1},\\
&a_{2,1} = \frac{2}{3}\,\frac{\%1}{2\,b_3\,c_3 - 1},\ b_2 = -\frac{3}{4}\,\frac{4\,b_3{}^2\,c_3{}^2 - 4\,b_3\,c_3 + 1}{\%1},\ c_2 = \frac{2}{3}\,\frac{\%1}{2\,b_3\,c_3 - 1},\\
&b_1 = \frac{1}{4}\,\frac{4\,b_3 - 12\,b_3\,c_3 - 1 + 12\,b_3\,c_3{}^2}{\%1}\},\\
&\{b_2 = 0,\ a_{3,2} = a_{3,2},\ a_{3,1} = \frac{2}{3} - a_{3,2},\ b_3 = \frac{3}{4},\ c_3 = \frac{2}{3},\ b_1 = \frac{1}{4},\ a_{2,1} = \frac{2}{9}\,\frac{1}{a_{3,2}},\ c_2 = \frac{2}{9}\,\frac{1}{a_{3,2}}\}\\
&\%1 := -1 + 3\,b_3\,c_3{}^2
\end{aligned}
$$

这样，对于 $s=3$, 我们得到两个 (参数化的) 解，可由下面的系数图表表示. 注意，在第一个解中，未知数 c_3 和 b_3 是自由参数，即它们可取任意值. 在解集中，用 $c_3 = c_3$ 和 $b_3 = b_3$ 表示. 对于第二个解， $a_{3,2}$ 是自由参数 (参见图 19.1). 如果令 $a_{3,2} = 2/3$, 由第一个解可得 3 阶的 Heun 方法.

图 19.1 3 阶 Runge-Kutta 方法的 3 步

$$
\begin{array}{c|ccc}
0 & & & \\
\frac{2}{3}\frac{3b_3c_3^2-1}{2b_3c_3-1} & \frac{2}{3}\frac{3b_3c_3^2-1}{2b_3c_3-1} & & \\
c_3 & \frac{1-6b_3c_3+12b_3^2c_3^3}{4b_3(3b_3c_3^2-1)} & \frac{2b_3c_3-1}{4b_3(3b_3c_3^2-1)} & \\
\hline
 & \frac{12b_3c_3^2-1+4b_3-12b_3c_3}{12b_3c_3^2-4} & \frac{3}{4}\frac{(2b_3c_3-1)^2}{1-3b_3c_3^2} & b_3
\end{array}
\qquad
\begin{array}{c|ccc}
0 & & & \\
\frac{2}{9a_{3,2}} & \frac{2}{9a_{3,2}} & & \\
\frac{2}{3} & \frac{2}{3} - a_{3,2} & a_{3,2} & \\
\hline
 & \frac{1}{4} & 0 & \frac{3}{4}
\end{array}
$$

现在我们将提出解方程组的另外两个方法.

19.3.1 Gröbner 基

计算机代数中，关于多项式的广泛多样的问题，可用多项式理想表示. 这类问题的一些例子就是关于 (多项式) 边关系或 (多项式) 方程组解的简化.

我们回忆一个理想的定义. 给定一个域 $\mathbf{K}$ 上的多变量多项式的 (有限) 集合 $F = \{f_1, f_2, \ldots, f_n\}$, 则定义由集合 F 生成的理想 $\langle F \rangle$ 是

$$
\langle F \rangle = \langle f_1, f_2, \ldots, f_n \rangle = \left\{ \sum_{i=1}^{m} h_i f_i | h_i \in \mathbf{K}[x_1, \ldots, x_n] \right\},
$$

即 $\langle F \rangle$ 是所有多项式的集合，这些多项式可由 F 中多项式的组合来构造. 称 F 中的元素为理想 $\langle F \rangle$ 的一个基.

Gröbner 基方法首先将给定的集合 F, 变换成标准基 G (称为一个 Gröbner 基), 使得 $\langle G \rangle = \langle F \rangle$, 然后，求解带有 $\langle G \rangle$, 而不是 $\langle F \rangle$ 的问题. 这个变换过程通过消去多项式集合的项来完成，它类似于 Gaussian 消去法的过程. 从一对旧的多项式 f_1, f_2 可形成新的多项式 $p = \alpha f_1 + \beta f_2$, 其中 α 和 β 是适当选取的多项式. 通过此过程，一个接一个地将变量消去 (根据特殊的顺序). 如果系统有有限个解，则最后得到一个变量的一个多项式，求解它，并代入 Gröbner 基的其它元素. 这种方法是 Gaussian 消去法的三角化过程的推广. 我们将不再详细讨论计算 Gröbner 基的理论和算法，作为入门，读者可参见 [1]. 然而，我们将利用在 MAPLE 中实现的 Gröbner 基算法 [3], 求解我们的方程组：

```
> G := grobner[gbasis](eqns,
```

```
>            [b[1],b[2],c[2],a[2,1],a[3,1],a[3,2],b[3],c[3]], plex);
```

$$G := [b_1 - 6\,{b_3}^2\,a_{3,\,2}\,c_3 + b_3 - 1 + 3\,b_3\,a_{3,\,2},\ b_2 + 6\,{b_3}^2\,a_{3,\,2}\,c_3 - 3\,b_3\,a_{3,\,2},\ c_2 - a_{2,\,1},\ 3\,a_{3,\,2}\,a_{2,\,1} + 6\,a_{3,\,2}\,b_3\,{c_3}^2 - c_3 - 2\,a_{3,\,2},\ 6\,a_{2,\,1}\,b_3\,c_3 - 3\,a_{2,\,1} - 6\,b_3\,{c_3}^2 + 2,\ a_{3,\,1} + a_{3,\,2} - c_3,\ 1 - 2\,b_3\,c_3 - 4\,b_3\,a_{3,\,2} + 12\,{b_3}^2\,{c_3}^2\,a_{3,\,2}]$$

最后的方程包含三个未知数，说明系统有两个自由参数．我们可从这最后的方程解出 $a_{3,2}$.

```
> a[3,2] := normal(solve(G[7], a[3,2]));
```

$$a_{3,\,2} := \frac{1}{4}\,\frac{2\,b_3\,c_3 - 1}{b_3\,(-1 + 3\,b_3\,{c_3}^2)}$$

如果分母不为零，则此表达式是唯一的解．让我们先假设是这种情况．通过 Gröbner 基 G 的回代过程，可得到其它未知数的值．

```
> a[3,1] := normal(solve(G[6], a[3,1]));
```

$$a_{3,\,1} := \frac{1}{4}\,\frac{-6\,b_3\,c_3 + 1 + 12\,{b_3}^2\,{c_3}^3}{b_3\,(-1 + 3\,b_3\,{c_3}^2)}$$

```
> a[2,1] := normal(solve(G[5], a[2,1]));
```

$$a_{2,\,1} := \frac{2}{3}\,\frac{-1 + 3\,b_3\,{c_3}^2}{2\,b_3\,c_3 - 1}$$

```
> normal(G[4]);
```

$$0$$

```
> c[2] := normal(solve(G[3], c[2]));
```

$$c_2 := \frac{2}{3}\,\frac{-1 + 3\,b_3\,{c_3}^2}{2\,b_3\,c_3 - 1}$$

```
> b[2] := factor(solve(G[2], b[2]));
```

$$b_2 := -\frac{3}{4}\,\frac{(2\,b_3\,c_3 - 1)^2}{-1 + 3\,b_3\,{c_3}^2}$$

```
> b[1] := normal(solve(G[1], b[1]));
```

$$b_1 := \frac{1}{4}\,\frac{4\,b_3 - 12\,b_3\,c_3 - 1 + 12\,b_3\,{c_3}^2}{-1 + 3\,b_3\,{c_3}^2}$$

我们得到与直接用 solve 命令相同的解．

对于 $b_3(3b_3c_3{}^2 - 1) = 0$, 将多项式 $3b_3{}^2c_3{}^2 - b_3$ 加到由后者 Gröbner 基的计算得到的理想上，可得另一个解．首先，必须对这些参数赋值．

```
> a := 'a':  b := 'b':  c := 'c':
> grobner[gbasis]([G[], 3*b[3]^2*c[3]^2-b[3]],
>            [b[1],b[2],c[2],a[2,1],a[3,1],a[3,2],b[3],c[3]], plex);
```

$$[4\,b_1 - 1,\ b_2,\ c_2 - a_{2,\,1},\ 9\,a_{3,\,2}\,a_{2,\,1} - 2,\ 3\,a_{3,\,1} + 3\,a_{3,\,2} - 2,\ -3 + 4\,b_3,\ 3\,c_3 - 2]$$

这对应于用 solve 命令计算的第二个解．

19.3.2 结式

因为两个多项式 $f, g \in \mathbf{R}[x]$ 的结式可消去 x, 即 $\text{res}_x(f,g) \in \mathbf{R}$, 所以用结式求解的过程也类似 Gaussian 消去法. 两个多项式 $f,g \in \mathbf{R}[x]$ 的结式 (写成 $\text{res}_x(f,g)$), 定义为 f 和 g 的 Sylvester 矩阵的行列式 (参见任何代数的入门书, 如 [13]). 下面的定理 (来自 [5]) 说明如何利用结式求解多项式方程组.

假设 $\bar{F}$ 是一个闭的代数域, 并假设

$$f = \sum_{i=0}^{m} a_i(x_2, \ldots, x_r)x_1^i, \; g = \sum_{i=0}^{n} b_i(x_2, \ldots, x_r)x_1^i,$$

是 x_1 中正次的 $\bar{F}[x_1, \ldots, x_r]$ 的元素. 如果 $(\alpha_1, \ldots, \alpha_r)$ 是 f 和 g 的公共零点, 则它们关于 x_1 的结式满足

$$\text{res}_{x_1}(\alpha_2, \ldots, \alpha_r) = 0.$$

反之, 如果 $\text{res}_{x_1}(\alpha_2, \ldots, \alpha_r) = 0$, 则下列条件之一成立:

$$\begin{aligned}
&a_m(\alpha_2, \ldots, \alpha_r) = b_n(\alpha_2, \ldots, \alpha_r) = 0, \\
&\forall x \in \bar{F} : f(x, \alpha_2, \ldots, \alpha_r) = 0, \\
&\forall x \in \bar{F} : g(x, \alpha_2, \ldots, \alpha_r) = 0, \\
&\exists \alpha_1 \in \bar{F}, \text{ 使得 } (\alpha_1, \alpha_2, \ldots, \alpha_r) \text{ 是 } f \text{ 和 } g \text{ 的一个公共零点},
\end{aligned}$$

其中最后一个条件是我们感兴趣的.

现在我们利用 MAPLE 的 **resultant** 函数, 将方程组变换成三角形式. 首先, 收集不同集合 B_j 中的方程, 使得

$$B_j = \{p \in \bar{F}[x_j, \ldots, x_r] - \bar{F}[x_{j+1}, \ldots, x_r]\}.$$

于是可把方程看成 $\mathbf{Q}[b_1, a_{3,1}, b_2, c_2, a_{2,1}, a_{3,2}, b_3, c_3]$ 中的元素, 并定义集合 B_j.

```
> X := [b[1],a[3,1],b[2],c[2],a[2,1],a[3,2],b[3],c[3]]:
> for i to nops(X) do B[i] := {} od:
> for p in eqns do
>    for i while not has(p, X[i]) do od;
>    B[i] := B[i] union {primpart(p)}
> od:
```

让我们考察集合 B_i. 特别地, 考虑每个集合中的元素个数.

```
> seq('B['.i.']' = B[i], i=1..nops(X));
```

$$\begin{aligned}
&B[1] = \{-b_3 - b_2 + 1 - b_1\}, B[2] = \{ \\
&\quad -3\,b_3\,a_{3,2}{}^2 - 3\,b_2\,a_{2,1}{}^2 - 6\,b_3\,a_{3,1}\,a_{3,2} + 1 - 3\,b_3\,a_{3,1}{}^2, \\
&\quad -2\,b_3\,a_{3,2} + 1 - 2\,b_2\,a_{2,1} - 2\,b_3\,a_{3,1}, -3\,b_2\,c_2\,a_{2,1} + 1 - 3\,b_3\,c_3\,a_{3,1} - 3\,b_3\,c_3\,a_{3,2}\}, \\
&\quad B[3] = \{-2\,b_3\,c_3 - 2\,b_2\,c_2 + 1, 1 - 3\,b_2\,c_2{}^2 - 3\,b_3\,c_3{}^2\}, B[4] = \{1 - 6\,b_3\,a_{3,2}\,c_2\}, \\
&\quad B[5] = \{-6\,b_3\,a_{3,2}\,a_{2,1} + 1\}, B[6] = \{\}, B[7] = \{\}, B[8] = \{\}
\end{aligned}$$

于是，从 B_3 中的两个元素可消去 b_2，给出集合 B_4 的另外一个元素，因此也有两个元素．由这两个方程消去 c_2，可得到 $\mathbf{Q}[a_{3,2}, b_3, c_3]$ 中的一个结式，将它加入 B_6.

```
> primpart(resultant(B[3][1], B[3][2], b[2]));
```

$$3\,{c_2}^2 - 6\,{c_2}^2\,b_3\,c_3 - 2\,c_2 + 6\,c_2\,b_3\,{c_3}^2$$

```
> B[4] := B[4] union {"}:
> primpart(resultant(B[4][1], B[4][2], c[2]));
```

$$1 - 2\,b_3\,c_3 - 4\,b_3\,a_{3,\,2} + 12\,{b_3}^2\,{c_3}^2\,a_{3,\,2}$$

```
> B[6] := B[6] union {"}:
```

从上面的计算可知，整个系统有两个自由参数，因此我们停止这个过程，并开始回代这个解．我们将仅构造第二个解，当 $4b_3(3b_3{c_3}^2 - 1 \neq 0)$ 时，它是有效的.

```
> a[3,2] := normal(solve(B[6][1], a[3,2]));
```

$$a_{3,\,2} := \frac{1}{4}\,\frac{2\,b_3\,c_3 - 1}{b_3\,(-1 + 3\,b_3\,{c_3}^2)}$$

```
> a[2,1] := normal(solve(B[5][1], a[2,1]));
```

$$a_{2,\,1} := \frac{2}{3}\,\frac{-1 + 3\,b_3\,{c_3}^2}{2\,b_3\,c_3 - 1}$$

```
> c[2] := normal(solve(B[4][1], c[2]));
```

$$c_2 := \frac{2}{3}\,\frac{-1 + 3\,b_3\,{c_3}^2}{2\,b_3\,c_3 - 1}$$

```
> b[2] := normal(solve(B[3][1], b[2]));
```

$$b_2 := -\frac{3}{4}\,\frac{(2\,b_3\,c_3 - 1)^2}{-1 + 3\,b_3\,{c_3}^2}$$

```
> a[3,1] := normal(solve(B[2][2], a[3,1]));
```

$$a_{3,\,1} := \frac{1}{4}\,\frac{-6\,b_3\,c_3 + 1 + 12\,{b_3}^2\,{c_3}^3}{b_3\,(-1 + 3\,b_3\,{c_3}^2)}$$

```
> b[1] := normal(solve(B[1][1], b[1]));
```

$$b_1 := \frac{1}{4}\,\frac{4\,b_3 - 12\,b_3\,c_3 - 1 + 12\,b_3\,{c_3}^2}{-1 + 3\,b_3\,{c_3}^2}$$

我们得到与用 Gröbner 基的相同结果.

19.4 完整的算法

在本节中，我们把所有用于获得 $s=3$ 的 Runge-Kutta 公式的 MAPLE 语句，编辑成一个称为 RungeKutta(s,m) 的 MAPLE 过程．它可计算任意 m 阶 s 阶段的 Runge-Kutta 方法的系数．不幸的是，对于 $s>3$ 的值，用 MAPLE 和其它计算机代数系统不能直接求解这些方程．多项式方程组已经变得太复杂，它可通过加上所谓的对称条件

$$c_i = \sum_{j=1}^{i-1} a_{i,j}, \quad i = 2..s,$$

简化．对于微分方程 $y'=1, y(0)=0$, 我们要求在 $x+c_ih$ 点的所有预测值 $y_i^* = y + h\sum_{j=1}^{i-1} a_{i,j}$, 应该与通过插入精确解 $y(x)=x$ 所得的值相同，这样可得到此对称条件．用这些附加的条件，可计算阶为 $m=1,\ldots,4$ 的 Runge-Kutta 公式．算法 19.1 是完整的 MAPLE 代码．

19.4.1 例 1

我们首先对参数 $s=2$ 和 $m=2$, 测试此过程

```
> RungeKutta(2, 2);
```

$$\{b_1 = 1 - b_2, b_2 = b_2, a_{2,1} = \frac{1}{2}\frac{1}{b_2}, c_2 = \frac{1}{2}\frac{1}{b_2}\}$$

我们又有了一个自由参数，即 b_2. 图 19.2 以表记号显示这个结果． 当 $b_2=1$ 时，得到改进的 Euler

图 19.2 一般 2 阶段 Runge-Kutta 方法

0		
$\frac{1}{2b_2}$	$\frac{1}{2b_2}$	
	$1-b_2$	b_2

方法 (参见方程 (19.3)). 当 $b_2=1/2$ 时，得到法则

$$y_{k+1} = y_k + \frac{h}{2}(f(x_k, y_k) + f(x_k + h, y_k + hf(x_k, y_k)))$$

这就是著名的 Heun 方法．

算法 19.1 RungeKutta 过程

```
RungeKutta := proc(s, m)
  local TaylorPhi, RungeKuttaPhi, d, vars, eqns, k, i, j;
  global a, b, c, h;
  # Taylor series
  D(y) := x -> f(x,y(x)):
  TaylorPhi := convert(taylor(y(x+h),h=0,m+1), polynom):
  TaylorPhi := normal((TaylorPhi - y(x))/h);
  # RK-Ansatz:
  c[1] := 0;
  for i from 1 to s do
```

```
    k[i] := taylor(f(x+c[i]*h, y(x) +
            sum(a[i,j]*k[j], j=1..i-1)*h), h=0, m):
  od:
  RungeKuttaPhi := 0:
  for i from 1 to s do
    RungeKuttaPhi := RungeKuttaPhi + b[i] * k[i]:
  od:
  RungeKuttaPhi := series (RungeKuttaPhi, h, m):
  RungeKuttaPhi := convert(RungeKuttaPhi, polynom);
  d := expand(TaylorPhi - RungeKuttaPhi):
  vars := {seq(c[i], i=2..s),
           seq(b[i], i=1..s),
           seq((seq(a[i,j], j=1..i-1)), i = 2..s)};
  eqns := {coeffs(d, indets(d) minus vars)};
  # symmetry condition:
  eqns := eqns union {seq(sum(a[i,'j'], 'j'=1..i-1) -
                          c[i], i=2..s)};
  solve(eqns, vars);
end:
```

19.4.2 例 2

在这个例子里，我们计算 4 阶的 4 阶段 Runge-Kutta 公式.

```
> RK4 := RungeKutta(4, 4):
> RK4[2];
```

$$\{b_2 = -b_4 + \frac{1}{6},\ a_{4,2} = -\frac{1}{12}\frac{1}{b_4},\ a_{4,3} = \frac{1}{3}\frac{1}{b_4},\ b_3 = \frac{2}{3},\ a_{3,1} = \frac{3}{8},\ a_{3,2} = \frac{1}{8},\ c_4 = 1,$$
$$a_{4,1} = \frac{1}{4}\frac{-1+4\,b_4}{b_4},\ c_3 = \frac{1}{2},\ a_{2,1} = 1,\ c_2 = 1,\ b_1 = \frac{1}{6},\ b_4 = b_4\}$$

我们得到两个解，其中一个解有一个自由参数 b_4，可用图 19.3 表示它.

另一个解有两个自由参数 c_2 和 $a_{3,2}$.% 项代表公共的子表达式，它被显示在解集的下面，*RootOf* 表达式代表二次方程的两个根

$$576(1 - 1c_2 - 2a_{3,2}c_2 + 4a_{3,2}{c_2}^2)x^2 + 24(c_2 - 2)x + 1.$$

```
> RK4[1];
```

$$\Big\{a_{2,1} = c_2,\ c_4 = 1,\ b_4 = \frac{1}{288}\frac{\%5}{(-c_2 + 1 + 4\,a_{3,2}\,{c_2}^2 - 2\,a_{3,2}\,c_2)\,\%1\,(c_2 - 1)},\ a_{4,2} = \frac{1}{2}(-1 + 2\,c_2$$
$$- {c_2}^2 - 168\,\%4 + 24\,\%1\,{c_2}^2 + 4\,a_{3,2}\,{c_2}^3 + 192\,{a_{3,2}}^2\,{c_2}^2\,\%1 - 576\,{a_{3,2}}^2\,{c_2}^3\,\%1$$
$$+ 384\,\%1\,{c_2}^4\,{a_{3,2}}^2 + 24\,\%1 + 264\,\%3 - 96\,\%2 - 48\,\%1\,c_2 + a_{3,2}\,c_2 - 5\,a_{3,2}\,{c_2}^2)\Big/($$

图 19.3 $s=4, m=4$ 的简单解

$$\begin{array}{c|cccc} 0 & & & & \\ 1 & 1 & & & \\ \frac{1}{2} & \frac{3}{8} & \frac{1}{8} & & \\ 1 & 1-\frac{1}{4b_4} & -\frac{1}{12b_4} & \frac{1}{3b_4} & \\ \hline & \frac{1}{6} & \frac{1}{6}-b_4 & \frac{2}{3} & b_4 \end{array}$$

$$a_{3,2}\,\%5\,c_2^2),\ a_{4,3}=12\,\frac{\%1\,(-c_2^2+2\,c_2-1+4\,a_{3,2}\,c_2^3-6\,a_{3,2}\,c_2^2+2\,a_{3,2}\,c_2)}{c_2\,a_{3,2}\,\%5},$$

$$b_2=-\frac{1}{576}\,\frac{96\,\%2-24\,\%1+1}{(c_2-1)\,\%1\,c_2^2\,a_{3,2}},\ b_3=\frac{\%1}{c_2\,a_{3,2}},\ a_{3,1}=-\frac{1}{24}\,\frac{-24\,\%1+1+24\,\%1\,a_{3,2}}{\%1},a_{4,1}=$$

$$\frac{1}{2}(1-2\,c_2+c_2^2-24\,\%1+24\,\%1\,c_2^3+96\,\%2+552\,\%4-408\,\%3+72\,\%1\,c_2$$
$$-72\,\%1\,c_2^2-192\,a_{3,2}^2\,c_2^2\,\%1+960\,a_{3,2}^2\,c_2^3\,\%1-1728\,\%1\,c_2^4\,a_{3,2}^2$$
$$+1152\,a_{3,2}^2\,c_2^5\,\%1-288\,\%1\,c_2^4\,a_{3,2}-a_{3,2}\,c_2+3\,a_{3,2}\,c_2^2)\Big/(a_{3,2}\,\%5\,c_2^2),$$

$$c_3=\frac{1}{24}\,\frac{-1+24\,\%1}{\%1},\ b_1=\frac{1}{576}\,\frac{288\,\%3-96\,\%2-24\,\%1\,c_2-1+24\,\%1}{\%1\,c_2^2\,a_{3,2}},\ c_2=c_2,$$

$$\left.a_{3,2}=a_{3,2}\right\}$$

$$\%1:=\mathrm{RootOf}(1+(-48+24\,c_2)_Z+(-576\,c_2+576+2304\,a_{3,2}\,c_2^2-1152\,a_{3,2}\,c_2)_Z^2)$$
$$\%2:=\%1\,c_2\,a_{3,2}$$
$$\%3:=\%1\,c_2^2\,a_{3,2}$$
$$\%4:=\%1\,c_2^3\,a_{3,2}$$
$$\%5:=120\,\%1\,c_2-48\,\%1-672\,\%3+192\,\%2-96\,\%1\,c_2^2+576\,\%4+2\,c_2-1$$

从这个解，如果要求三个参数 $a_{3,1}, a_{4,1}$ 和 $a_{4,2}$ 都为零，则可得 4 阶古典的 Runge-Kutta 方法. 即如果令 $c_2=1/2$ 和 $a_{3,2}=1/2$, 并取多项式 $288x^2-36x+1$ 的根 $x=1/12$, 则可得到它.

```
> solve(subs(RK4[1], {a[3,1]=0, a[4,1]=0, a[4,2]=0}));
```

$$\{c_2=\frac{1}{2},\ a_{3,2}=\frac{1}{2}\}$$

```
> subs(", RK4[1]);
```

$$\Big\{c_4=1,\ a_{2,1}=\frac{1}{2},\ b_4=\frac{1}{6},\ a_{3,1}=-\frac{1}{24}\,\frac{-12\,\%1+1}{\%1},\ b_3=4\,\%1,\ a_{4,3}=1,\ b_2=\frac{1}{36}\,\frac{1}{\%1},$$

$$a_{4,2}=-\frac{1}{3}\,\frac{-\frac{3}{8}+\frac{9}{2}\,\%1}{\%1},\ \frac{1}{2}=\frac{1}{2},\ a_{4,1}=-\frac{1}{3}\,\frac{\frac{3}{8}-\frac{9}{2}\,\%1}{\%1},\ c_3=\frac{1}{24}\,\frac{-1+24\,\%1}{\%1},$$

$$\left.b_1=\frac{1}{72}\,\frac{-1+24\,\%1}{\%1}\right\}$$

$$\%1:=\mathrm{RootOf}(1-36\,_Z+288\,_Z^2)$$

```
> allvalues(%1);
```

$$\frac{1}{24}, \frac{1}{12}$$

```
> subs(%1=1/12, "");
```

$$\{a_{4,1} = 0,\, a_{4,2} = 0,\, b_2 = \frac{1}{3},\, a_{3,1} = 0,\, b_3 = \frac{1}{3},\, c_4 = 1,\, a_{2,1} = \frac{1}{2},\, b_4 = \frac{1}{6},\, a_{4,3} = 1,\, \frac{1}{2} = \frac{1}{2},\, c_3 = \frac{1}{2},\, b_1 = \frac{1}{6}\}$$

如果令 $c_2 = 1/3$ 和 $a_{3,2} = 1$, 则得到两个其它的 Runge-Kutta 方法，其中之一是，根据它的权 b_i, 对应多项式 $256x^2 - 40x + 1$ 的根 1/8 的方法，称为 3/8 法则.

```
> subs(c[2]=1/3, a[3,2]=1, RK4[1]);
```

$$\left\{1 = 1,\, a_{2,1} = \frac{1}{3},\, \frac{1}{3} = \frac{1}{3},\, a_{4,3} = -\frac{32}{3}\,\frac{\%1}{-8\,\%1 - \frac{1}{3}},\, b_4 = -\frac{3}{256}\,\frac{-8\,\%1 - \frac{1}{3}}{\%1},\right.$$

$$a_{4,2} = \frac{9}{2}\,\frac{-\frac{14}{27} + \frac{176}{27}\,\%1}{-8\,\%1 - \frac{1}{3}},\, a_{3,1} = -\frac{1}{24}\,\frac{1}{\%1},\, b_1 = \frac{1}{64}\,\frac{16\,\%1 - 1}{\%1},\, c_3 = \frac{1}{24}\,\frac{-1 + 24\,\%1}{\%1},$$

$$\left. a_{4,1} = \frac{9}{2}\,\frac{\frac{4}{9} - \frac{160}{27}\,\%1}{-8\,\%1 - \frac{1}{3}},\, b_2 = \frac{3}{128}\,\frac{8\,\%1 + 1}{\%1},\, b_3 = 3\,\%1,\, c_4 = 1\right\}$$

$$\%1 := \mathrm{RootOf}(1 - 40\,_Z + 256\,_Z^2)$$

```
> allvalues(%1);
```

$$\frac{1}{32}, \frac{1}{8}$$

```
> subs(%1=1/8, "");
```

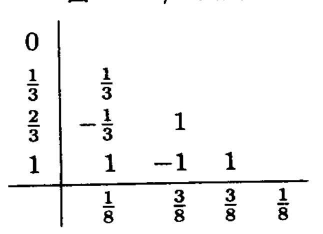
$$\{1 = 1,\, a_{2,1} = \frac{1}{3},\, \frac{1}{3} = \frac{1}{3},\, c_4 = 1,\, a_{4,3} = 1,\, b_4 = \frac{1}{8},\, a_{3,1} = \frac{-1}{3},\, a_{4,2} = -1,\, b_2 = \frac{3}{8},\, b_3 = \frac{3}{8},\, a_{4,1} = 1,\, c_3 = \frac{2}{3},\, b_1 = \frac{1}{8}\}$$

这两个方法的系数如下所示:

图 9.4 古典 Runge-Kutta 方法

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

图 9.5 3/8 法则



0				
$\frac{1}{3}$	$\frac{1}{3}$			
$\frac{2}{3}$	$-\frac{1}{3}$	1		
1	1	-1	1	
	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$

19.5 结论

我们已经说明，为了得到 Runge-Kutta 显示公式定义的方程，MAPLE 是非常有用的. 注意，我们没有试图简化这些方程. 简化技术是著名的. 我们构造的方程是强制处理公式的结果，从而，只能得到不超过 4 阶的 Runge-Kutta 公式. 更高阶的方程仍然太大，以致用今天的计算机代数系统无法求解.

早在 1966 年，Moses 就曾猜想，用一个计算机代数系统不能建立对应 5 阶 5 阶段 Runge-Kutta 方法的方程组著名的不相容性 [12]. 对今天的计算机代数系统，这仍然有效. 因此，必须改进求解多项式方程组的算法. 一个研究方向是利用方程的对称性.

这一章也证明了，为了构造直到 10 阶 (17 阶段) 的 Runge-Kutta 公式 [7], 数学家努力简化方程组是正确的.

参考文献

[1] B. BUCHBERGER, *Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory*, in Progress, directions and open problems in multidimensional systems theory, ed. N. K. Bose, D. Reidel Publishing Co, 1985, pp. 189-232.

[2] J. C. BUTCHER, *The non-existence of ten Stage eight Order Explicit Runge-Kutta Methods*, BIT, 25,1985, pp. 521-540.

[3] S. CZAPOR AND K. GEDDES, *On Implementing Buchbergers's Algorithm for Gröbner Bases*, ISSAC86, 1986, pp 233-238.

[4] G. E. COLLINS, *The Calculation of Multivariate Polynomial Resultants*, Journal of the ACM, 18, No. 4,1971, pp. 512-532.

[5] K. O. GEDDES, S. R. CZAPOR AND G. LABAHN, *Algorithms for Computer Algebra*, Kluwer, 1992.

[6] G. H. GONNET AND M. B. MONAGAN, *Solving Systems of Algebraic Equations, or the Interface between Software and Mathematics*, Computers in Mathematics, Conference at Stanford University, 1986.

[7] E. HAIRER, S. P. NØRSETT AND G. WANNER, *Solving Ordinary Differential Equations I*, Springer-Verlag Berlin Heidelberg, 1987.

[8] R. J. JENKS, *Problem # 11: Generation of Runge-Kutta Equations*, SIGSAM Bullatin, 10, No. 1, 1976, pp. 6.

[9] W. KUTTA, *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen*, Zeitschrift für Math. u. Phys., 46, 1901, pp. 435-453.

[10] M. MONAGAN AND J. S. DEVITT, *The D Operator and Algorithmic Differentiation*, Maple Technical Newsletter, No. 7, 1992.

[11] M. MONAGAN AND R. R. RODINI, *An Implementation of the Forward and Reverse Modes of Automatic Differentiation in Maple*, Proceedings of the Santa Fe conference on Computational Differentiation, SIAM, 1996.

[12] J. MOSES, *Solution of Systems of Polynomial Equations by Elimination*, Comm. of the ACM, 9, No. 8, 1966, pp. 634-637.

[13] B. L. VAN DER WAERDEN, *Algebra I*, Springer-Verlag, Berlin, 1971.

第二十章 双相半波整流器的瞬时反应

H.J. Halin and R. Strebel

20.1 引言

电路是由线性微分方程组所描述的典型例子，这些方程的系数为常数或随时间而变化．在时间域里对这类系统进行数值模拟是十分困难的，尤其当系统非常大且具有广泛分布的离散特征值时．分析电路的方法已经有好几本参考书介绍，如 [2]．市场上也有大量的计算机辅助分析的软件包，其程序往往沿用诸如 ECAP[3] 或 SPICE2[5] 的代码．这些程序完全使用数值计算方法，因而也就有了一些知名的支持者和反对者．

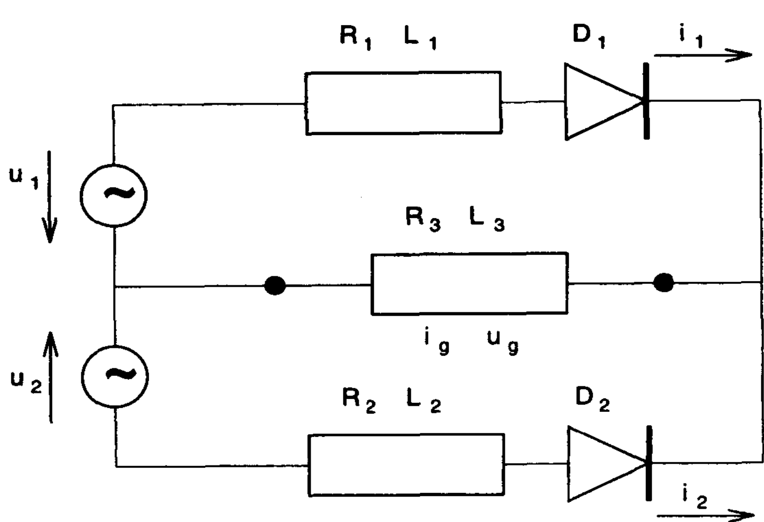
下面将用 MAPLE V[1] 特有的分析能力和精度来求解一个小的但是很有技巧的电路问题．它能大致地说明，为什么模拟双相半波整流器的瞬时反应这种问题，在许多方法中都是需要很高技巧的问题．这一点后面会有更详细的介绍．

光是运用数学模型的常规程序来求数值解是不够的．必须借助于某些技巧以克服下面将会讨论的一些数值计算中的困难．这就是为什么缺乏经验的分析者往往不能在进行这类仿真研究时马上成功的原因．

20.2 问题概述

将要研究的问题是：当两个交流电源作为外部驱动函数放入电路以后，模拟双相半波整流器的电流瞬时反应．这个问题最早在文献 [4] 中被介绍．模型的结构如图 20.1 所示．电路的组成是：两个理想的二极管 D_1 和 D_2，两个电阻和两个电抗，各为 R_1 、 R_2 、 L_1 和 L_2，两个开路的电源 $u_1(t)$ 、 $u_2(t)$，负载则用电阻 R_3 和电抗 L_3 表示．流经二极管的电流分别为 $i_1(t)$ 和 $i_2(t)$，$i_g(t)$ 是整流后的电流， $u_g(t)$ 是整流后的电压．时间用 t 表示．

图 20.1 整流系统的结构



根据二极管的导通状态可分为三种不同的情况. 具体运用哪一种取决于与时间相关的变量 $diod_1$ 和 $diod_2$ 的逻辑值. 以后我们会定义这些变量, 但我们要强调的是, 使用它们只是为了叙述方便. 在后面的程序列表中, 这些变量并未显示地表示出来.

1. 仅有二极管 D_1 导通. 即 $diod_1 = true$ 和 $diod_2 = false$:

$$\frac{\mathrm{d}}{\mathrm{d}t}i_1(t) = \frac{u_1 - i_1 R_{13}}{a_2} \tag{20.1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}i_2(t) = 0 \tag{20.2}$$

2. 仅有二极管 D_2 导通. 即 $diod_1 = false$ 和 $diod_2 = true$:

$$\frac{\mathrm{d}}{\mathrm{d}t}i_1(t) = 0 \tag{20.3}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}i_2(t) = \frac{u_2 - i_2 R_{23}}{a_1} \tag{20.4}$$

3. 二极管 D_1 和二极管 D_2 都导通. 即 $diod_1 = true$ 和 $diod_2 = true$:

$$\frac{\mathrm{d}}{\mathrm{d}t}i_1(t) = \frac{a_1 u_1 - L_3 u_2 - z_1 i_1 - z_2 i_2}{b} \tag{20.5}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}i_2(t) = \frac{a_2 u_2 - L_3 u_1 - z_3 i_2 - z_4 i_1}{b} \tag{20.6}$$

其中

$$\begin{aligned}
a_1 &= L_2 + L_3\\
a_2 &= L_1 + L_3\\
b &= L_1 L_2 + L_1 L_3 + L_2 L_3\\
R_{13} &= R_1 + R_3\\
R_{23} &= R_2 + R_3\\
z_1 &= a_1 R_1 + L_2 R_3\\
z_2 &= L_2 R_3 - L_3 R_2\\
z_3 &= a_2 R_2 + L_1 R_3\\
z_4 &= L_1 R_3 - L_3 R_1
\end{aligned}$$

注意任何时候这三种情况只有一种成立, 而第四种情况即两个变量都是 $false$ 是没有意义的, 除非是电源没有连结的稳态情况. 对 $diod_1$ 和 $diod_2$ 的值引入有关的条件以后, 我们便可讨论在任一给定时刻与这三种情况哪一种有关的必要条件.

整流电压 $u_g(t)$ 和整流电流 $i_g(t)$ 分别为

$$u_g(t) = R_3(i_1 + i_2) + L_3\left(\frac{\mathrm{d}}{\mathrm{d}t}i_1(t) + \frac{\mathrm{d}}{\mathrm{d}t}i_2(t)\right) \tag{20.7}$$

$$i_g(t) = i_1 + i_2. \tag{20.8}$$

电压 $u_1(t)$ 和 $u_2(t)$ 写作

$$u_1(t) = U\sin(\omega t) \tag{20.9}$$

$$u_2(t) = -u_1(t). \tag{20.10}$$

其中

$$U = \sqrt{2}U_{eff} \tag{20.11}$$

而且

$$\omega = 2\Pi\nu. \tag{20.12}$$

其中 U_{eff} 是有效电压而 ν 是频率.

为了说明哪个二极管是导通的，我们已经引入了两个逻辑变量 $diod_1$ 和 $diod_2$. 在任何时刻 t, 这些变量的逻辑值由以下条件控制:

$$diod_1 = \begin{cases} true & \text{如果 } i_1 > 0 \text{ 或者 } u_1 > u_g, \\ false & \text{反之} \end{cases} \tag{20.13}$$

$$diod_2 = \begin{cases} true & \text{如果 } i_2 > 0 \text{ 或者 } u_2 > u_g, \\ false & \text{反之} \end{cases} \tag{20.14}$$

现在研究在时间间隔为 $t_{start} \leq t \leq t_{final}$ 期间的瞬时状态，此处 t_{start} 和 t_{final} 代表模拟时间段的上端值和下端值.

研究中会用到以下值:

$$R_1 = 2[\Omega] \tag{20.15}$$

$$R_2 = 2[\Omega] \tag{20.16}$$

$$R_3 = 10[\Omega] \tag{20.17}$$

$$L_1 = 0.04[H] \tag{20.18}$$

$$L_2 = 0.04[H] \tag{20.19}$$

$$L_3 = 0.20[H] \tag{20.20}$$

$$U_{eff} = 100[V] \tag{20.21}$$

$$\nu = 50[Hz]. \tag{20.22}$$

选择初始条件为

$$i_1(0) = 0 \tag{20.23}$$

$$i_2(0) = 0. \tag{20.24}$$

20.3 应用常规编程和软件包的困难

使用 Runge-Kutta 一类的积分算法来求解上述问题，分析人员必须编写主程序或编写调用积分子程序的过程. 这些调用包含在 DO 循环之中，是在积分程序控制之下运行，直到积分程序完成了在独立变量的指定区间内的积分. 当程序控制回到主程序时，就得到上一段区间的解并把它输出. 然后，用上一段区间解的值作为初值，再在下一段区间重新调用积分子程序.

在积分过程中，积分子程序要调用由分析人员提供的‘函数赋值’的过程. 这时，独立变量 t 的当前值，大量的一阶常微分方程和在时刻 t 的解向量，都被传递给相关过程. 这些过程基本都包含微分方程的代码，即方程 (20.1-20.5) 以及 (20.15) 等等式所给出的常数，它们是主程序引入的局部变量. 然后才可能去计算微分方程右边的值，并把含时间的解向量返回给积分子程序.

在给函数赋值之前，还必须从 (20.13) 确定变量 $diod_1$ 的 $diod_2$ 的逻辑值，以便决定应用三种情况中的哪一种两个常微分方程的系统. 由于要用 (20,13) 的 u_g 值来求 $diod_1$ 的 $diod_2$ 的值，显而易见出现了一种隐含的关系. 为了计算 $\frac{d}{dt}i_1$ 和 $\frac{d}{dt}i_2$ 的值，必须知道整流电压 u_g. 然后才能找出 $diod_1$ 和 $diod_2$, 最后可以从给定的情况得到 $\frac{d}{dt}i_1$ 和 $\frac{d}{dt}i_2$.

为了解决这个困难，Jentsch 在 [4] 中提出，使用延迟 τ 秒以后的 u_g 值，τ 是一个时间小量. 在 (20.13) 中使用的延迟值 u_{gd} 为

$$u_{gd}(t) = \begin{cases} u_g(t-\tau) & \text{如果 } t > \tau \\ u_g(0) & \text{如果 } t \le \tau \end{cases}$$

注意这等价于

$$u_{gd}(s) = e^{(-\tau s)}u_g(s)$$

也就是在运用标准的工程技术时，对线性问题运用 Laplace 变换的结果.

在数值计算中 $u_{gd}(t)$ 是靠在 $u_g(t_i)$ 和 $u_g(t_{i+1})$, $(i = 1, 2, \ldots)$ 之间的插值来求得的，其中 $(t_i \le t - \tau \le t_{i+1})$. $u_g(t_i)$ 和 $u_g(t_{i+1})$ 各是第 $i-1$ 和 i 个积分步骤 μ_g 的值.

在 $t_0 = t_{start}$ 开始的第一步积分,还产生了另一个困难. 因为两个初始条件 (20.23), 即 $i_1(t_0)$,$i_2(t_0)$, 两个激发条件，即 $u_1(t_0), u_2(t_0)$ 和 $u_g(t)$ 全部为零，这样导致 $diod_1 = diod_2 = false$, 因此也就无法确定应用哪一种情况.

甚至任意假定一种情况有效，也会导致 $\frac{d}{dt}i_1(t_0) = \frac{d}{dt}i_2(t_0) = 0$. 显然，从物理的角度来说，至少应该有一个二极管导通，因为在 $t = t_0$ 时，$u_1(t)$ 有一个正的导数，而 $u_2(t)$ 有一个负的导数. 这意味着这两个逻辑值中至少有一个为真. 因而在 $t = t_0$ 时，决定 $diod_1$ 和 $diod_2$ 的值必须根据 $u_1(t), u_2(t), u_g(t)$ 的导数值.

在给出了 $diod_1$ 和 $diod_2$ 的初值以后，也能决定 $t_0 + \epsilon$ 的其它位置它们的值，这里 ϵ 是一个小的正数.

20.4 用 MAPLE 求解

用 MAPLE V[1] 求解这个例子，可以巧妙地避开上面提到的数值困难.

下面介绍一个程序，它用来计算整流器的瞬时反应. 这个程序由三个文件组成，分别是 `halfwave.map`, `data.map` 和 `procs.map`.

我们的程序是从文件 `halfwave.map` 开始. 重新启动后，我们要在 `halfwave.map` 中读入后面会用到的 `unassign` 库程序. 然后再读入 `data.map` 和 `procs.map`. 接着要给出 t_{start} 和 t_{final} 值以说明变量 t 的范围.

halfwave.map 中的调用序列

```
readlib (unassign):
read ('data.map');
read ('procs.map');
tstart := 0:
tfinal := 0.050:
i := rectify_solve (array(1..2,[0,0]), tstart..tfinal):
```

```
ig := unapply(i[1](t)+i[2](t),t):
ug := unapply(R.3*ig(t) + L.3*diff(ig(t),t),t):

plot (i[1], tstart..tfinal, title='Plot 1: i1(t)');
plot (i[2], tstart..tfinal, title='Plot 2: i2(t)');
plot (ig, tstart..tfinal, title='Plot 3: ig(t)');
plot (ug, tstart..tfinal, title='Plot 4: ug(t)');
```

微分方程的积分是从调用 `rectify_solve` 开始，这个程序作为主程序列在 `procs.map` 的末尾．只要知道 `rectify_solve` 返回了两个解析解 $i_1(t)$ 和 $i_2(t)$ 作为二维矢量 `ddata` 的元素就行了．只要每个元素本身可以用列表表示，解析解是可以分段求解的．注意调用 `rectify_solve` 时，两个状态变量的初始值和模拟时间的范围也要输入．在本章最后会介绍在 `halfwave.map` 末尾的作图．

注意在方程 (20.1-20.24) 中使用的下标在程序中被表示成园点加下标，如方程中的 R_3 在程序中被标记成 R.3.

在文件 data.map 中，外部驱动函数 $u_1(t), u_2(t)$ 是预先给定的．此后，问题中某些必要的参数也要引入．在文件中 `nof_cases` 和 `nof_states` 分别代表了在整流器工作时会遇到的各种不同情况和各种状态变量．

data.map 中微分方程的解

```
#------- input

u[1] := t ->  U*sin(Omega*t):
u[2] := t -> -U*sin(Omega*t):

#------- parameters

U := sqrt(2)*100:
nu := 50:
Omega := 2*Pi*nu:

R.1 := 2:
R.2 := 2:
R.3 := 10:
L.1 := convert(0.04,rational):
L.2 := convert(0.04,rational):
L.3 := convert(0.2,rational):
a.1 := L.2 + L.3:
a.2 := L.1 + L.3:
b := L.1*L.2 + L.1*L.3 + L.2*L.3:
R.13 := R.1 + R.3:
```

```
R.23 := R.2 + R.3:
z.1 := a.1*R.1 + L.2*R.3:
z.2 := L.2*R.3 - L.3*R.2:
z.3 := a.2*R.2 + L.1*R.3:
z.4 := L.1*R.3 - L.3*R.1:

#------- equations

nof_cases := 3:
nof_states := 2:

vars := {seq(j[k](t), k=1..nof_states)}:
deqn := array (1..3, [
  { diff(j[1](t),t) = j[1](t)*(-R.13/a.2) + u[1](t)/a.2,
    diff(j[2](t),t) = 0,
    j[1](t0) = j10,
    j[2](t0) = j20
  },
  { diff(j[1](t),t) = 0,
    diff(j[2](t),t) = j[2](t)*(-R.23/a.1) + u[2](t)/a.1,
    j[1](t0) = j10,
    j[2](t0) = j20
  },
  { diff(j[1](t),t) = j[1](t)*(-z.1/b) + j[2](t)*(-z.2/b)
                 + 1/b*(a.1*u[1](t) - L.3*u[2](t)),
    diff(j[2](t),t) = j[1](t)*(-z.4/b) + j[2](t)*(-z.3/b)
                 + 1/b*(a.2*u[2](t) - L.3*u[1](t)),
    j[1](t0) = j10,
    j[2](t0) = j20
  }]):

#------- global params

eps := 1/nu*1.0e-6; # get some 'relative' gap
Digits := 14;
_EnvsignumO := 0;

#------- solve the system for all cases

unassign ('k','t0','j10','j20'):
```

```
for k from 1 to nof_cases do
  dsol[k] := simplify (dsolve (deqn[k], vars)):
  assign (dsol[k]);
  for n from 1 to nof_states do
    i[k][n] := unapply (j[n](t), t, t0, j10, j20):
  od:
  unassign ('j[1](t)','j[2](t)'):
od:
```

随后是三种可能情况对应的三套微分方程公式，以及它们的形式解．注意迄今为止，无论是起始时间 t0 还是初始条件 j10 和 j20 都不是数值形式．

为了寻找在某一时刻 t 状态变量 $i_1(t), i_2(t)$ 的根．必须提供一个误差范围如 $\epsilon = 1.0e-6$. 由于三种情况都要用到的时间间隔的长度都以某种方式依赖于外部驱动函数 $u_1(t)$ 和 $u_2(t)$ 的波长 $1/\nu$, 更合适的做法是引入"相对"误差范围 $\epsilon = 1.0e-6/\nu$.

最后一个文件是 procs.map, 它包含了"主程序"以及数值计算中的某些过程和函数.

procs.map 中定义的过程

```
#------- Utilities

`diff/piecewise` := proc ()
  description `cheap diff on piecewise expression.`,
              `Maple V4's library function considers\
               discontinuities, but is _much_ slower.`;
  local g, s, n, k, t;
  g := [args[1..nargs-1]];
  t := args[nargs];  n := nops(g);
  s := seq(op([g[2*k-1],diff(g[2*k],t)]),k=1..floor(n/2));
  if (type(n,odd)) then s := s,diff(g[n],t); fi;
  RETURN (piecewise(s));
end:

signum_rightof := proc (f, t::numeric)
  description `return signum of f(t+)`;
  global eps;
  local r, s, x;
  if (type (t, rational)) then
    s := simplify (series (f(x), x=t));
    r := op(1,s);
    RETURN (signum(r));
  fi;
  RETURN (signum(evalf (f(t+eps))));
```

```
end:

fsolve_smallest := proc (f, r::range)
  description `return smallest zero in [r]. NULL if none.`;
  global eps;
  local tfrom, tto, s, sprev, t;
  tfrom := op(1,r); tto := op(2,r);
  sprev := NULL;
  while (tfrom <= tto) do
    s := fsolve (f(t), t, t=tfrom..tto);
    if (whattype(s) <> float) then RETURN(sprev); fi;
    if ((s < tfrom) or (tto < s)) then RETURN(sprev); fi;
    tto := s-eps;
    sprev := s;
  od;
  RETURN(sprev);
end:

#------- get next interval

case[1,0] := 1:  case[0,1] := 2:  case[1,1] := 3:
# i[2] = 0:  i[1] = 0:

next_case := proc (t0::numeric, j::array)
  description `returns case that follows point t0.`;
  global eps, case;
  RETURN (case[signum_rightof(j[1],t0), signum_rightof(j[2],t0)]);
end:

next_interval := proc (t0::numeric, i0::array, VAR_case)
  description `returns currents for next interval.`,
              `In VAR_case next case.`;
  local t, j, s, case;
  for s from nof_cases by -1 to 1 do
    j := array(1..2);
    j[1] := unapply (i[s][1](t,t0,i0[1],i0[2]), t);
    j[2] := unapply (i[s][2](t,t0,i0[1],i0[2]), t);
    case := next_case (t0, j);
    if (s = case) then VAR_case := s; RETURN (j); fi;
  od;
```

```
  ERROR (`Continuation failed`);
end:

crit_smallest := proc (f, r::range(numeric))
  description `smallest zero of f in ]r] (without left border).`,
              `right(r) if (f == 0) or (f(t) <> 0) in r.`;
  global eps;
  local s, tfrom, tto;
  tfrom := op(1,r); tto := op(2,r);
  if (f = 0) then RETURN (tto); fi;
  s := fsolve_smallest (f, tfrom+eps..tto);
  if (whattype(s) <> float) then RETURN (tto); fi;
  RETURN (s);
end:

next_crit := proc (j::array, r::range(numeric),
                   case::integer, VAR_i)
  description `returns next critical point.`,
              `In VAR_i current i[] at this point.`;
  local t, ug, ud, crit, tcrit, i;
  ug := unapply (evalf(R.3*(j[1](t) + j[2](t)) +
        L.3*(diff(j[1](t),t)+diff(j[2](t),t))), t);
  ud[1] := unapply (u[1](t) - ug(t), t);
  ud[2] := unapply (u[2](t) - ug(t), t);
  if (case = 1) then
    crit[1] := crit_smallest (j[1], r);
    crit[4] := crit_smallest (ud[2], r);
    tcrit := min(crit[1], crit[4]);
  elif (case = 2) then
    crit[2] := crit_smallest (j[2], r);
    crit[3] := crit_smallest (ud[1], r);
    tcrit := min(crit[2], crit[3]);
  elif (case = 3) then
    crit[1] := crit_smallest (j[1], r);
    crit[2] := crit_smallest (j[2], r);
    crit[3] := crit_smallest (ud[1], r);
    crit[4] := crit_smallest (ud[2], r);
    tcrit := min(crit[1], crit[2], crit[3], crit[4]);
  fi;
  i := array (1..2, [j[1](tcrit), j[2](tcrit)]);
```

```
  if (crit[1] = tcrit) then i[1] := 0; fi;
  if (crit[2] = tcrit) then i[2] := 0; fi;
  VAR_i := i;
  RETURN (tcrit);
end:

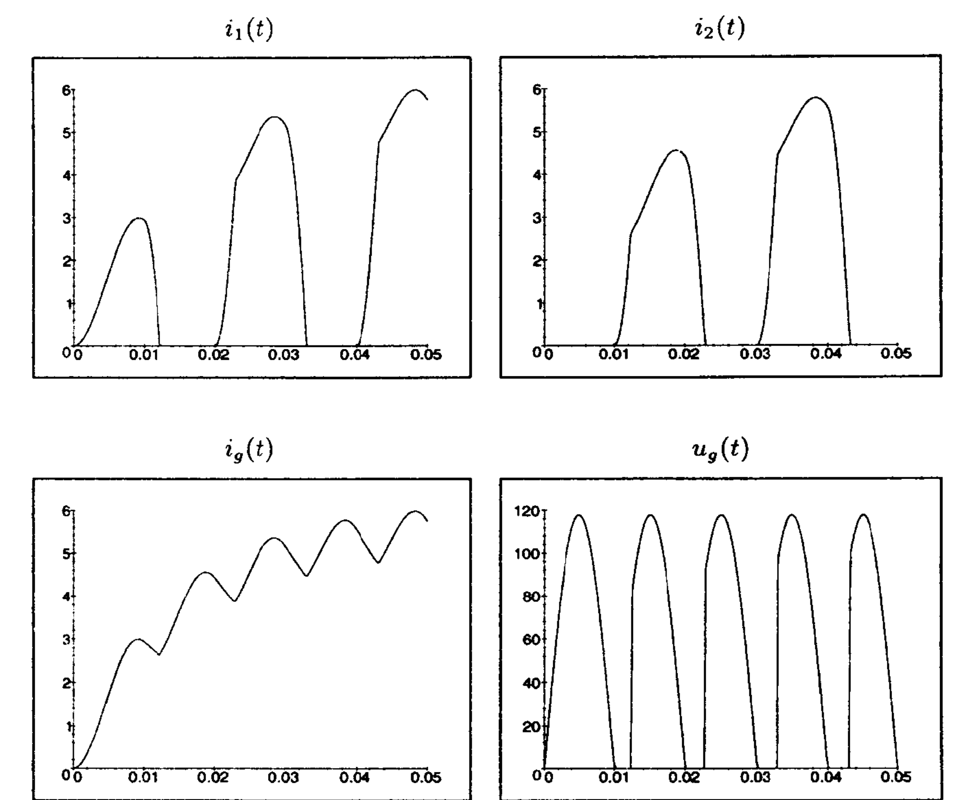
#------- main program

rectify_solve := proc (i0::array(numeric), r::range(numeric))
  description 'returns i[1..2] as piecewise functions';
  global nof_states, eps;
  local tfrom, tto, tcur, scur, res, i, j, k, ires, n, nof, d, s, t;
  tfrom := op(1,r); tto := op(2,r);
  res := array (1..nof_states);
  for k from 1 to nof_states do res[k] := NULL; od;
  tcur := tfrom; i := i0;
  while (tcur < tto) do
    unassign ('scur');
    j := next_interval (tcur, i, scur);
    for k from 1 to nof_states do
      res[k] := res[k], [tcur, j[k], scur];
    od;
    unassign ('i');
    tcur := next_crit (j, tcur..tto, scur, i);
  od;
  ires := array (1..nof_states);
  for k from 1 to nof_states do
    nof := nops([res[k]]);
    d := res[k],[tto+eps];
    s := seq(op([d[n][1] <= t and t < d[n+1][1],d[n][2](t)]),n=1..nof);
    ires[k] := unapply(piecewise(s),t);
  od;
  RETURN (ires);
end:
```

procs.map 中的“主程序”是控制大多数执行过程的程序. 该程序使用了状态变量的初始值和在每个间隔开始时独立变量的值, 在每个间隔中三种情况中仅有一种成立. 通过连续地调用 next_interval 过程, 就可以确定首先采用三种情况中的哪一种. 在这个过程中, next_case 也要被用到, 它本身又要调用 sign_rightof. 当然在文件 halfwave_map 中已经找到的解析解自始至终都要用到. 正如在 sign_rightof 中看到的那样, 以解析的形式确定在时刻 $t = t_{start}$ 使用哪一种情况, 这要通过计算状态变量 $i_1(t), i_2(t)$ 高阶导数的值, 直到求出导数不为零的阶为止. 否则, 就要找到某个位置

$t+\epsilon$ 的状态， t 是某次情况改变的最后位置．一旦所采用的情况确定了，处理结果的信息将存储在“主程序”的列表中．此后，再考虑下一个间隔，直到独立变量的全部范围都包括在内．

下面几个图画出了在整个过程中电流 $i_1(t), i_2(t)$，整流电流 $i_g(t)$ 和整流电压 $u_g(t)$ 的图形．



参考文献

[1] B. W. CHAR, K. O. GEDDES, G. H. GONNET, B. L. LEONG, M. B. MONAGAN AND S. M. WATT, *Maple Language/Reference Manual*, Springer Verlag, New York, 1991.

[2] L. O. CHUA AND P. M. LIN, *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques*, Prentice-Hall, Englewood Cliffs, N.J.,1975.

[3] R. W. JENSEN AND M. D. LIEBERMAN, *IBM Electronic Circuit Analysis Program*, Prentice-Hall, Englewood Cliffs, N.J.,1968.

[4] W. JENTSCH, *Digitale Simulation kontinuierlicher Systeme*, Oldenbourg Verlag, München und Wien, 1969.

[5] L. W. NAGEL, *SPICE2: A Computer Program to Simulate Semiconductor Circuits*, University of California, Ph.D. Dissertation, Berkeley, California, 1975.

第二十一章　输电设备中的电路

J. Waldvogel

21.1　引言

在过去的几年里，具有内部关断能力的大功率半导体器件已经开始使用. 这种器件叫门开关 (GTO) 闸流二极管，它有几个硅层，每层发挥一定的作用；它们能够在毫秒的时间内在数千伏的电压下关断 1000 安培的电流. 常见的电阻，电抗，电容和闸流二极管可以一起组合在输电设备的电路中，闸流二极管可简单地看成一个开关. 这种技术仍然是一个很活跃的研究领域，它有许多很重要的应用，比如交直流转换 (双向), 机车和电车的速度控制，以及供电网络和电站的控制.

闸流二极管开关的每一个确定状态都遵循 Kirchhoff 定律，这种电路的动力学行为可用具有常系数的线性常微分方程组来描述，这里假定电路元件都是线性的. 如果开关改变其状态则电路也改变其结构，但是开关之前电路中电流的最终状态决定了开关之后的初始条件.

因而，如果开关的时间可以忽略，输电设备中电路的数学模型，就是一组具有分段常系数的线性微分方程. 我们假定电路的动力学问题可以被 n 个时间 t 的连续函数来描述，它由相关变量组成的向量 $\mathbf{x}(t) \in \mathbf{R}^n$ 来描述. 用矩阵符号来表示，该模型可以写为

$$\dot{\mathbf{x}} = A(t)\mathbf{x} + \mathbf{p}(t) \tag{21.1}$$

在此, 园点代表对时间求导，$A(t)$ 是阶跃函数, 也就是一个给定分段常数的 $n \times n$ 矩阵, 而 $\mathbf{p}(t) \in \mathbf{R}^n$ 是一个给定的驱动函数. 在交流电 (AC) 的情况下，$\mathbf{p}(t)$ 和 $A(t)$ 常常是周期函数. 不失一般性，这周期可以规一化为 2π, 从 $\mathbf{p}(t)$ 的 Fourier 分解来看，我们将把一次谐波作为模型. 通常，解 $\mathbf{x}(t)$ 是用初始条件 $\mathbf{x}(0) = \mathbf{x}_0$ 来表述的，但是其它的表述如 $\mathbf{x}(t)$ 的周期性会在 21.3 中考虑.

图 21.1　简化的 SVC 电路

SVC 电路的描述

变量	描述
$U_R(t), U_S(t), U_T(t)$	电站产生的交流电压
R_M, L_M	主要部分的电阻和电感
R_C, L_C, C	SVC 的电阻，电感和电容
$I_R(t), I_S(t), I_T(t)$	主要部分的注入电流
$U(t)$	加在 SVC 直流电容两端的电压

具有一个常数矩阵 A 的线性微分方程组的初值问题是一个初等微积分的课题 (见 [1]), 可以用 A 的特征值和特征向量来处理，或者利用矩阵指数 e^{At} 来处理. 即使矩阵 $A(t)$ 是分段常数，初始值问题的显示解也可直接求出，尽管在 $A(t)$ 有许多不连续点时，求解可能相当繁琐.

可以看到， MATLAB 的特点使我们能够很好地构造这个初值问题的解 $\mathbf{x}(t)$. 周期解也很容易计算和作图. 在本章使用电力网络控制领域的一种称之为 SVC (Static Var Compensator) 的特殊设备，来说明 MATLAB 在输电线路中的应用. 这里 Var 代表 Volt-Ampere reactive.

在输电网络中使用 SVC 是为了补偿电压的下降，它是由于电力线路的损失和用户负载的变化所造成. 在通常的三相交流电系统中，每个周期需要六个开关动作，以便在每个周期相应的时间间隔中，使电力反冲脉冲从一个相转到另一个相.

图 21.1 表示了一个简化的但包括了主要部分的 SVC 电路，闸流二极管由开关来代表. 为了简化，没有考虑电网的负载. 每相电路在任何时刻只可以关闭一个开关. 对图 21.1 的电路选择 $n=3$ 个独立变量即可，分别是

$$x_1(t) = U(t), \quad x_2(t) = I_R(t), \quad x_3(t) = \frac{1}{\sqrt{3}}(I_S(t) - I_T(t)) \tag{21.2}$$

电路的动力学问题由方程 (21.1) 描述，其中

$$\mathbf{p}(t) = \frac{1}{L}(0, \cos t, \sin t)^T, \quad L = L_M + L_C, \quad R = R_M + R_C \tag{21.3}$$

和

$$A(t) = B(\varphi(t)), \quad B(\varphi) = \begin{bmatrix} 0 & \frac{1}{C}\cos\varphi & \frac{1}{C}\sin\varphi \\ -\frac{2}{3L}\cos\varphi & -\frac{R}{L} & 0 \\ -\frac{2}{3L}\sin\varphi & 0 & -\frac{R}{L} \end{bmatrix}. \tag{21.4}$$

开关角 $\varphi(t)$ 是一个给定的分段常量的函数，它控制闸流二极管的动作. 在 SVC 的六个脉冲中它被选择为

$$\varphi(t) = \frac{\pi}{3}\mathrm{round}\left(\frac{3}{\pi}(t-\tau)\right), \tag{21.5}$$

其中，移动量 τ 是 SVC 的一个参数，且 $|\tau| \le R$. 为使交流电路的周期 (常常是 0.02 秒) 成为 2π, 使用了规一化单位来表示参量 C, L 和 R, 它们的典型值分别是

$$C = 0.2, \quad L = 0.15, \quad R = 0.005. \tag{21.6}$$

实际上， C, L 和 R 的量级分别是 150nF,2Hy 和 20Ω. 进一步的技术细节请参见文献 [2,3].

21.2 具有分段常系数的线性微分方程组

我们考虑未知函数 $\mathbf{x}(t)\in\mathbf{R}^n$ 的微分方程组 (21.1)

$$\dot{\mathbf{x}}=A(t)\mathbf{x}+\mathbf{p}(t),$$

为了简单，假定给定的 2π 周期的驱动函数 $\mathbf{p}(t)$ 只有一次谐波

$$\mathbf{p}=\mathbf{b}\mathrm{e}^{\mathrm{i}t}+\bar{\mathbf{b}}\mathrm{e}^{-\mathrm{i}t},\quad \mathbf{b}\in\mathbf{C}^n. \tag{21.7}$$

使用复函数符号可大大简化方程，MATLAB 也完全支持复函数. 假定分段常数实矩阵 $A(t)$ 也具有 2π 的周期. 因而，(可能的)$m+1$ 个 $A(t)$ 的不连续值 t_k(跳跃点) 可如下引入

$$0=t_0<t_1<t_2<\ldots<t_{m-1}<t_m=2\pi,$$

而矩阵的离散值标记为

$$A(t)=A_k \qquad \text{其中}\ \ t_k\le t<t_{k+1},\quad k=0,\ldots,m-1. \tag{21.8}$$

给定初始条件为 $\mathbf{x}(0)=\mathbf{x}_0$, 方程 (21.1) 具有唯一解 $\mathbf{x}(t)$; 它在跳跃点的值记为

$$\mathbf{x}_k:=\mathbf{x}(t_k),\quad k=0,\ldots,m.$$

首先，我们在 k 个小间隔 $t\in[t_k,t_{k+1}]$ 中构造 (21.1) 的隐性解 $\mathbf{x}(t)$, 它满足的微分方程和初始条件是

$$\left.\begin{array}{rcl}\dot{\mathbf{x}}(t)&=&A_k\mathbf{x}(t)+\mathbf{p}(t),\ \ t\in[t_k,t_{k+1}]\\ \mathbf{x}(t_k)&=&\mathbf{x}_k\end{array}\right\}\quad k=0,\ldots,m-1. \tag{21.9}$$

通常作如下分解

$$\mathbf{x}(t)=\mathbf{y}(t)+\mathbf{z}(t), \tag{21.10}$$

其中，$\mathbf{z}(t)$ 为适当选择的特解；而 $\mathbf{y}(t)$ 为同构问题的解，满足

$$\dot{\mathbf{y}}(t)=A_k\mathbf{y}(t),\ \ \mathbf{y}(t_k)=\mathbf{x}_k-\mathbf{z}(t_k). \tag{21.11}$$

使用熟悉的矩阵指数可得到

$$\mathbf{y}(t)=\mathrm{e}^{A_kt}\mathbf{c}_k, \tag{21.12}$$

$\mathbf{c}_k$ 由 (21.11) 的第二个方程决定.

矩阵指数的计算是一个具有很长历史的重要问题，综述文章 [4]"计算矩阵指数的 19 种不确定方法" 清楚地表明了这一点. 在 MATLAB 的 expm 命令中，应用了最小不确定度方法 —Padé 近似法. 几乎在所有的情况下，它都能可靠、准确并快速地工作. 不过对 $n\times n$ 矩阵，它是一种耗时费力的运算，需要高达 $30n^3$ 次的运算.

如果矩阵 B 是对角化的，e^B 可以更容易计算. 利用特征值因子分解 $B=TDT^{-1},D=\mathrm{diag}(\lambda_1,\ldots,\lambda_n)$

$$\mathrm{e}^B=T\mathrm{e}^DT^{-1},\ \ \mathrm{e}^D=\mathrm{diag}(\mathrm{e}^{\lambda_1},\ldots,\mathrm{e}^{\lambda_n}), \tag{21.13}$$

其中 λ_j 是 B 的第 j 个特征值.

在 (21.4) 定义的矩阵 $A(t)=B(\varphi(t))$ 的情况下，得出的特征值与 φ 无关，可以明确地写为

$$\lambda_1=-r,\quad \lambda_{2,3}=-r/2\pm \mathrm{i}\omega \tag{21.14}$$

这里

$$r=-R/L,\quad \omega=\sqrt{\frac{2/3}{LC}-\frac{R^2}{4L^2}}.$$

因而，对任意选择的开关时间 t_k 来说， $\lambda_1,\lambda_2,\lambda_3$ 也是所有矩阵 A_k 的特征值．从具有正交矩阵

$$S(\varphi)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\varphi) & -\sin(\varphi)\\ 0 & \sin(\varphi) & \cos(\varphi)\end{bmatrix}$$

的相似关系

$$B(\varphi)=S(\varphi)B(0)S(\varphi)^{-1}$$

可以看出这一点．

因而可得到

$$\mathrm{e}^{B(\varphi)}=S(\varphi)\mathrm{e}^{B(0)}S(\varphi)^{-1},$$

其优点是 $\mathrm{e}^{B(0)}$ 是一个简明的表达，因为 $B(0)$ 具有块对角结构为：

$$\mathrm{e}^{B(0)}=\mathrm{e}^{r/2}\begin{bmatrix}\cos(\omega)I+\sin(\omega)/\omega B_0 & 0\\ 0 & \mathrm{e}^{r/2}\end{bmatrix}.$$

其中 I 是 2×2 单位矩阵， B_0 是 $B(0)$ 的左上部 2×2 块矩阵， r 和 ω 如上定义．请读者用 MAPLE 自行证明上述关系．使用 $\mathrm{e}^{B(\varphi)}$ 可能有 30 个基本操作和 6 个函数调用 (例如 exp 或 sin).

为了在 k 个时间段内完成解的构造，必须选择 (21.9) 的特解 $\mathbf{z}(t)$. 最简单的选择是与 $\mathbf{p}(t)$ 同频率的谐振运动，即

$$\mathbf{z}(t)=-\mathbf{u}_k\mathrm{e}^{\mathrm{i}t}-\bar{\mathbf{u}}_k\mathrm{e}^{-\mathrm{i}t} \tag{21.15}$$

其中，确定复向量 $\mathbf{u}_k\in\mathbf{C}^n$ 时必须满足 (21.9). 将 (21.15) 代入 (21.9) 式，得到条件

$$(A_k-\mathrm{i}I)\mathbf{u}_k=\mathbf{b}, \tag{21.16}$$

它是在 $\mathbf{C}$ 中关于 $\mathbf{u}_k$ 的一组线性方程，其中 I 是 $n\times n$ 的单位矩阵．因而形如 (21.15) 的解存在的必要和充分条件是 $\det(A_k-\mathrm{i}I)\neq 0$, 即 $\pm\mathrm{i}$ 必须不是 A_k 的特征值．从 (21.14) 得出，只要

$$R\neq 0 \text{ 或 } \frac{2}{3}\frac{L}{C}-\frac{R^2}{4}\neq L^2, \tag{21.17}$$

(对 (21.6) 给出的特殊数据成立,) 它就能被满足．共振情况 (条件 (21.17) 被破坏) 可通过在 (21.15) 中增加如 $\mathbf{v}_k t\mathrm{e}^{\mathrm{i}t}$ 的项来解决，但这种情况以后不再提及．

最后，结合 (21.10),(21.12) 和 (21.15), 得到的显示解

$$\mathbf{x}(t)=\mathrm{e}^{A_k t}\mathbf{c}_k-2\mathrm{Re}(\mathbf{u}_k\mathrm{e}^{\mathrm{i}t}),\quad t\in[t_k,t_{k+1}] \tag{21.18}$$

其中

$$\mathbf{u}_k=(A_k-\mathrm{i}I)^{-1}\mathbf{b} \tag{21.19}$$

和

$$\mathbf{c}_k = \mathrm{e}^{-A_k t_k}(\mathbf{x}_k + 2\mathrm{Re}(\mathbf{u}_k \mathrm{e}^{\mathrm{i}t_k})), \tag{21.20}$$

是 $t = t_k$ 时从 (21.18) 中得到. 把 $t = t_{k+1}$ 代入 (21.18) 得到下一跳跃点的 $\mathbf{x}(t)$ 值

$$\mathbf{x}_{k+1} = \mathrm{e}^{A_k t_{k+1}} \mathbf{c}_k - 2\mathrm{Re}(\mathbf{u}_k \mathrm{e}^{\mathrm{i}t_{k+1}}). \tag{21.21}$$

为了在多个点计算 $\mathbf{x}(t)$, 最好按照 (21.19),(21.20) 和 (21.21) 在一个循环中对所有 $k = 0, \ldots, m-1$ 预先计算并存储 $\mathbf{u}_k, \mathbf{c}_k$ 和 $\mathbf{x}_{k+1}$. 然后由 (21.18) 得到 $\mathbf{x}(t)$, 它至多涉及一个矩阵指数.

21.3 周期解

在如 SVC 一类的技术应用中, 人们感兴趣的是相应微分方程的周期解. 然而, 只有当周期解是吸引的时, 它才具有实际重要性, 它们在吸引域中经过长时间以后从任意初始状态自然产生.

在如 (21.1) 的线性问题中, 叠加原理是成立的. 因而周期解的稳定性是由 $\mathbf{p}(t) = 0$ 或 $\mathbf{b} = 0$ 定义的相应齐次问题决定的 (见方程 21.7). 从方程 (21.19), (21.20),(21.21) 的 $\mathbf{b} = 0$ 并将 $\mathbf{x}_k$ 用 $\mathbf{y}_k$ 代替, 由于矩阵 $A_k t_{k+1}$ 和 $A_k t_k$ 可交换, 我们得到

$$\mathbf{y}_{k+1} = \mathrm{e}^{A_k(t_{k+1} - t_k)} \mathbf{y}_k. \tag{21.22}$$

这里, $\mathbf{y}_k$ 表示齐次方程在跳跃点 t_k $(k = 0, \ldots, m)$ 的一个解的值. 因此在经过整个周期 $t_m = 2\pi$ 后 $\mathbf{y}_m$ 的值由线性图

$$\mathbf{y}_m = M\mathbf{y}_0, \tag{21.23}$$

其中

$$M = \prod_{k=0}^{m-1} \mathrm{e}^{A_k(t_{k+1} - t_k)} \tag{21.24}$$

(乘积是从右向左进行的) 是所谓的单值矩阵. 如果 $|\mu_j| < 1$ 对所有的 M 的特征值 μ_j 成立, 那么 (21.1) 的周期解全是吸引的.

由矩阵 (21.4) 和开关函数 (21.5) 给出的 6 脉冲 SVC 的特征值 μ_j 是与位移 τ 无关的. 在例 (21.6) 中得到的值是

$$\mu_1 = 0.84311362558494 \tag{21.25}$$

$$\mu_{2,3} = -0.77497080502505 \pm 0.42379742896324\mathrm{i} \tag{21.26}$$

因此如果周期解存在, 它是全局吸引的.

为了构造 $\mathbf{x}(t) = \mathbf{x}_P(t)$ 那样的解, 初始值 $\mathbf{x}_0 \in \mathbf{R}^n$ 必须求出, 使得在 (21.21) 的符号中有

$$\mathbf{x}_m = \mathbf{x}_0 \text{ 或 } \mathbf{f}(\mathbf{x}_0) := \mathbf{x}_m - \mathbf{x}_0 = 0. \tag{21.27}$$

由于问题的线性特点, (21.27) 定义的向量值函数 $\mathbf{f}$ 本身也是线性的. 因此能够计算出为了定义线性方程组 (21.27) 所需的 $\mathbf{f}$ 的 $n+1$ 个值. 这样做是必要的, 因为 $\mathbf{f}$ 的定义没有直接利用在 21.2 节末尾介绍的很复杂的算法.

$n+1$ 个点的值常常被方便地选为原点和 n 个单位点， $\mathbf{e}_j=(0,\ldots,0,1,0,\ldots,0)^{\mathrm{T}}$, 这里非零分量的位置为 j $(j=1,\ldots,n)$. 如果采用符号

$$\mathbf{f}_0:=\mathbf{f}(0),\mathbf{f}_j:=\mathbf{f}(\mathbf{e}_j),\quad(j=1,\ldots,n) \tag{21.28}$$

线性函数 $\mathbf{f}(\mathbf{x})$ 可显式表达为

$$\mathbf{f}(x)=\mathbf{f}_0+\sum_{j=1}^{n}(\mathbf{f}_j-\mathbf{f}_0)x_j \tag{21.29}$$

其中 $\mathbf{x}=(x_1,\ldots,x_n)^{\mathrm{T}}$. 初始值 $\mathbf{x}_0$ 满足 $\mathbf{f}(\mathbf{x}_0)=0$ 是从线性系统

$$F\mathbf{x}_0=\mathbf{f}_0 \tag{21.30}$$

得到的，其中矩阵 F 为

$$F=[\mathbf{f}_0-\mathbf{f}_1,\ldots,\mathbf{f}_0-\mathbf{f}_n]. \tag{21.31}$$

如果 F 是正规的，则存在唯一的周期解，在数值例子 (21.6) 中我们得到，对所有的 $\tau\in[-R,R]$, $\mathrm{cond}(F)\doteq 11.74$, 因而在本例中 F 决不是一个奇异矩阵.

21.4 应用 MATLAB

在本节我们将提供一个完整的 MATLAB 程序来计算下面介绍的 6 脉冲 SVC 的例子. 它基于上一段提出的具有分段常系数的 $n=3$ 的微分方程的显示解.

(a) 在 21.3 节中讨论的周期解 $\mathbf{x}_P$ 被生成，它在 $A(t)$ 的 $m+1$ 个跳跃点的值被存储. 可能的近退化可被 $\mathrm{cond}(F)$ 检测.

(b) 为了讨论 $\mathbf{x}_P$ 的稳定性，计算了与 $\mathbf{x}_P$ 有关的单值矩阵 M 以及它的特征值.

(c) $\mathbf{x}_P$ 的列表和作图 (密集输出)

(d) 周期解 $\mathbf{x}_P$ 的 Fourier 分析.

程序尽可能地保持通用性，尽管特殊例子的某些特点必然出现. 这使读者能把程序用到其它任何涉及到具有常系数线性常微分方程组的问题中去. 编程的主要目标是高效和简单，没有输入和输出的浪费. 就程序的简短而言，简明和高度的可靠性是达到了.

算法 21.1 函数 matrix

```
function [A] = matrix(k)
% generates the matrix A[k], k=0,...,m-1
%
  global m C L R

  phi = k*2*pi/(m-1);
  A   = [
                0, cos(phi)/C, sin(phi)/C
    -cos(phi)/1.5/L,      -R/L,          0
    -sin(phi)/1.5/L,         0,       -R/L
  ];
```

```
end % matrix
```

程序的核心是函数 f=solution(x0), 它在算法 21.2 中给出. 它解出具有初始向量 $\mathtt{x0} = \mathbf{x}_0$ 的 (21.2) 并返回由 (21.27) 定义的函数 $\mathtt{f} = \mathbf{f}(\mathbf{x}_0)$ 的值. 在实际调用中按照输出变量的数目 nargout(MATLAB 的永久变量), 矩阵 xx,uu,M 也被计算. 本例中的特定矩阵 A_k(见方程 (21.4),(21.5)) 是由算法 21.2 中的函数 A=matrix(k) 产生的. 为方便起见, 在表 21.1 中列出了被传递的全局变量.

表 21.1 全局变量的描述

变量	描述
m,n	维数参数
C, L, R	系统参数
b	方程 (21.7),(21.3) 中的非齐次 $\mathbf{b}$
tt(1:m+1)	跳跃点数组, $\mathtt{tt}(1+\mathtt{k}) = t_k$, $(k = 0\ldots m)$

算法 21.2 solution 函数

```
function [f, xx, uu, M] = solution(x0)
%SOLUTION       Solves the SVC problem over a period.
%               f == 0 <=> periodic solution
%
  global m n tt b

  if (nargout > 1), uu = []; xx = x0; end;
  if (nargout > 3), M  = eye(n); end;
  x = x0;
  for k = 1:m,
    A = matrix(k-1);
    E = expm(A*(tt(k+1)-tt(k)));
    u = (A-i*eye(n))\b;
    x = E*(x+2*real(u*exp(i*tt(k)))) ...
          - 2*real(u*exp(i*tt(k+1)));
    if (nargout > 1), uu = [uu,u]; xx = [xx,x]; end;
    if (nargout > 3), M  = E*M; end;
  end;
  f = x - x0;
end % solution
```

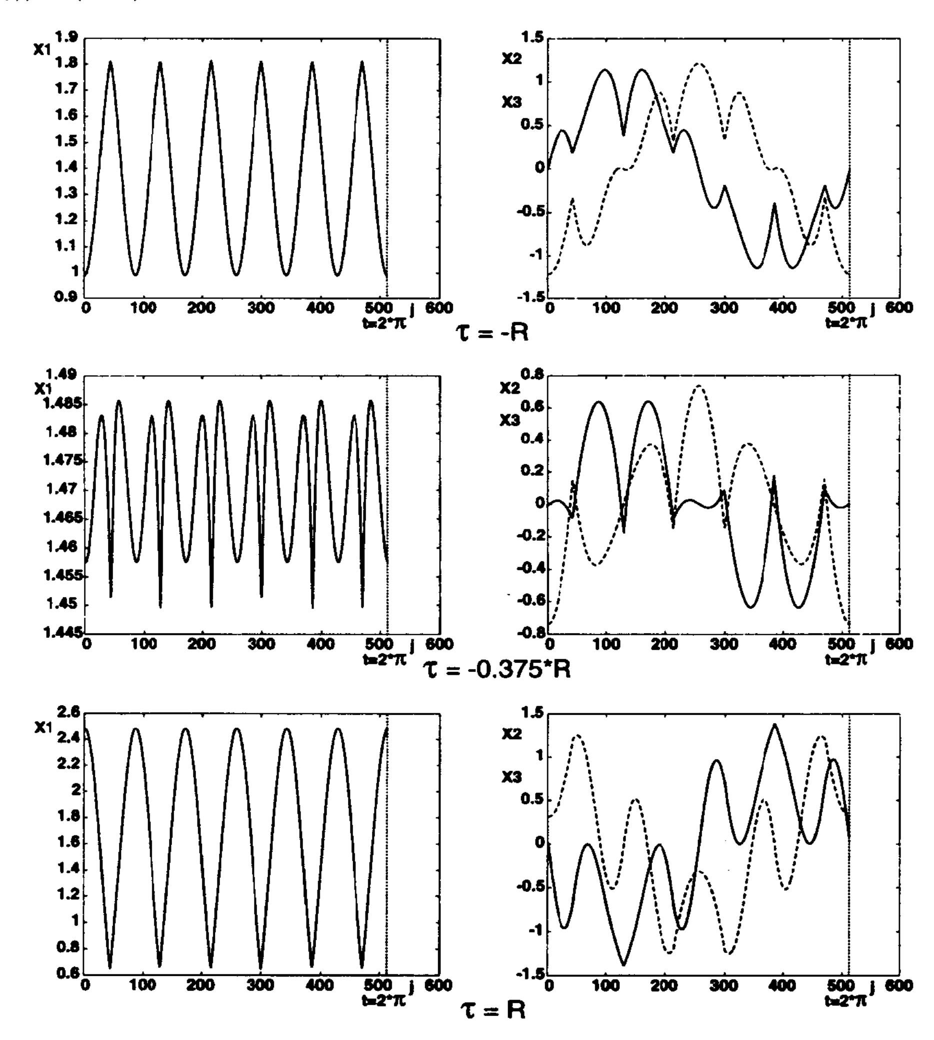
在 solution 的输出方面, xx 将包含 $\mathbf{x}_k$ 的值, uu 包含中间结果 $\mathbf{u}_k$, M 为单值矩阵. 这些变量在 solution 的第一行被初始化. 在随后的 k 循环中我们首先通过调用函数 matrix 产生矩阵 A_{k-1}. 方程 (21.19),(21.20) 和 (21.21) 如 21.1 节末尾那样求值. 仅有的修改是 (21.21) 的 $\mathbf{c}_k$ 被 (21.20) 所替代, 这减少了一半要计算的矩阵指数. 因而, 至少可以避免存储没有价值的向量 $\mathbf{c}_k$.

因为 MATLAB 的循环指数必须大于等于 1, 循环指标 k 每次变动 1. 存储新计算的向量 $\mathtt{u} = \mathbf{u}_k, \mathtt{x} = \mathbf{x}_{k+1}$ 是将它们分别附加在矩阵 uu,xx 上去. 在同一个循环中, 部分乘积 M 按照 (21.24) 被

更新. 最后的语句定义了函数 (21.27) 的值.

图 21.2 SVC 中的电压和电流

参数: $C = 0.2, L = 0.15, R = 0.005$, 由方程 (21.6) 得到. 三种情况的偏移 $\tau = -R, -0.375R, R$ 分别被显示. 左边的图画出电压 $x_1(t) = U(t)$ 对时间 t 的变化, 右边的图画出电流 $x_2(t) = I_S(t)$(实线) 和 $x_3(t) = (I_S(t) - I_T(t))/\sqrt{3}$(虚线).



在算法 21.3 中, 给定的主程序 per.m 通过调用函数 solution 执行了 (a) 到 (d) 的任务. 在初始化某些全局变量以后, 移动量 τ 的输入是通过程序请求来实现的. 用这种方法, 电路对 τ 的依赖性的敏感程度可以很容易地研究. 在后面的语句中, 跳跃点 tt(1:m+1) 的数组 (指数移动为 1) 定义为:

$$\begin{aligned}\mathtt{tt(1)} &= 0\\ \mathtt{tt(1+k)} &= t_k = \tau + (k - \tfrac{1}{2})\frac{\pi}{3}, \quad (k = 1, \ldots, 6)\\ \mathtt{tt(8)} &= 2\pi\end{aligned}$$

(参见 (21.5)). 然后周期解 x_P 与其初始值 x0 = x_0 一起被计算. 程序严格遵守方程 (21.28) 到 (21.31), 这是十分明显的.

算法 21.3 per 草稿

```
global m n C L R tt b

m = 7; n = 3;
C = 0.2; L = 0.15; R = 0.005;
b = [0;1;-i]/2/L;
N = 512;

tau = input('tau = ');
tt  = [0, 2*pi/(m-1)*[1/2:(m-3/2)]+tau, 2*pi];

f0 = solution(zeros(n,1));
F = []; for x = eye(n),
  F = [F, f0 - solution(x)];
end;
x0 = F\f0;
[ff, xx, uu, M] = solution(x0);

delta = 2*pi/N;
k = 0; xtab = [];
for j = 0:N-1,
  t = j*delta;
  if (tt(k+1) <= t),
    k = k+1;
    A = matrix(k-1);
    aux = expm(A*(t-tt(k))) ...
          *(xx(:,k)+2*real(uu(:,k)*exp(i*tt(k))));
    E = expm(A*delta);
  else
    aux = E*aux;
  end;
  xtab = [xtab, aux-2*real(uu(:,k)*exp(i*t))];
end;

figure(1); plot(xtab(1,:)');
figure(2); plot(xtab(2:3,:)');

cc = fft(1/N*xtab'); cc(1:32,:)
```

在下一段的程序里, $\mathbf{x}_P(t)$ 按步长 $\Delta = 2\pi/N$ 列表:

$$\mathtt{xtab(:,j)} = \mathbf{x}_P((j-1)\Delta), \quad j = 1, \ldots, N.$$

按照随后的快速 Fourier 分析的观点, N 必须是 2 的幂次. 如果 $N >> m$, 算法必须重新组织以便更有效地计算. 给定一个 t 值, 指数 k 要使 $t \in [t_k, t_{k+1}]$. 如果 $t \in [t_k, t_{k+1})$ 是间隔中的第一个取值点, $\mathbf{x}_P(t)$ 必须按照 (21.18) 计算. 因此我们开始计算的 (21.18) 第一项是

$$\mathbf{aux} := e^{A_k t}\mathbf{c}_k = \mathrm{e}^{A_k(t-t_k)}(\mathbf{x}_k + 2\mathrm{Re}(\mathbf{u}_k \mathrm{e}^{\mathrm{i}t_k})),$$

其中向量 $\mathbf{x}_k$ 和 $\mathbf{u}_k$ 分别从数组 xx 和 uu 中取值. 下一步是计算矩阵 E :=$\mathrm{e}^{A_k\Delta}$ 并将它存储在这一点, 这样可避免在同一个间隔以后的点中频繁地重复计算它. 此外, 它也能够按 aux:= E*aux 更新 aux.

在程序的最后一段中, $\mathbf{x}_P(t)$ 的第一分量, 即 $U(t)$(电容器 C 两端的电压) 是相对于 $j = 1+t/\Delta$ 画出的, 如 MATLAB 的图 1. 第二和第三个分量都是电流变量, 同时画在 MATLAB 的图 2 中. 最后, 利用 MATLAB 的命令 `fft` 对 $\mathbf{x}_P$ 的三个分量分别作了 Fourier 分析, 它需要列向量作为它的参数. 矩阵 cc 供打印 $\mathbf{x}(t)$ 的三个分量的前 32 个复 Fourier 系数之用.

在三种情况 $\tau = -R, \tau = -0.375R$ 和 $\tau = R$ 下的数据 (21.6) 产生了图 21.2. 对 τ 的微小变化的是十分敏感的. 理想的情况是, 电流 $x_2(t), x_3(t)$ 是正弦的. 这个领域的一个研究目标是: 降低在周期解 $\mathbf{x}_P$ 中的高次谐波引起的扰动.

21.5 结论

工艺过程的计算机模拟是过程设计和优化的有用的研究工具. 上面这个特定的电路模拟程序比一般的模拟程序在同样的问题上处理速度快十倍. 它的结果对数学模型的精确解有很高的近似程度 (14 位小数). 这使使用者能得到可靠的直到高阶的周期解的频谱, 反过来它又可以使设计者消去某些不需要的谐波.

显然, 这次仿真计算的成功很大程度上依靠 MATLAB 软件的高性能. 然而要得到最好的结果必须靠优秀的软件以及纯熟的数学分析技巧.

参考文献

[1] M. BRAUN, *Differential Equation and their Applications, Fourth ed.*, Springer, New York, 1993, 578pp.

[2] M. MEYER, *Leistungselektronik*, Springer, Berlin, 1990, 349pp.

[3] T. J. E. MILLER, *Reactive Power Control in Electric Systems*, J. Wiley and Sons, New York, 1982, 381pp.

[4] C. B. MOLER AND C. F. VAN LOAN, *Nineteen Dubious Ways to Compute the Exponential of a Matrix*, SIAM Review 20, 1978, pp. 801-836.

第二十二章 Newton 和 Kepler 定律

S.Bartoň

22.1 引言

这一章的目的是展示计算机代数在物理教学中的应用. 与 Newton 引力理论有关的七个例题将用 MAPLE 解决. 所有的例题只应用到著名的 Newton 引力定律

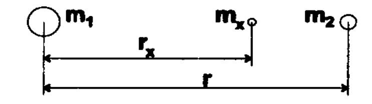
$$F = G\frac{m_1 m_2}{r^2}$$

或相应的势能公式.

22.2 二力的平衡

我们从简单的问题开始: 求位于两个质量分别为 m_1 和 m_2 的质点间的平衡点, 两点的距离为 r,(见图 22.1).

图 22.1 问题的定义




```
> R1 := G*m[1]*m[x]/rx^2 = G*m[x]*m[2]/(r - rx)^2;
```

$$R1 := \frac{G\, m_1\, m_x}{rx^2} = \frac{G\, m_x\, m_2}{(r - rx)^2}$$

```
> Sr := solve(R1, rx);
```

$$Sr := \frac{1}{2}\,\frac{(2\, m_1 + 2\sqrt{m_1\, m_2})\, r}{m_1 - m_2},\ \frac{1}{2}\,\frac{(2\, m_1 - 2\sqrt{m_1\, m_2})\, r}{m_1 - m_2}$$

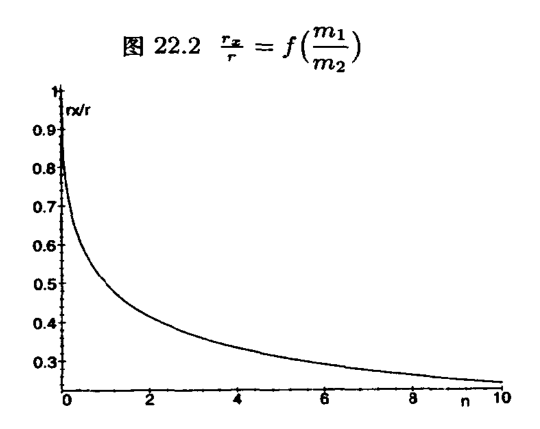
```
> rx := factor(Sr[2]):
> rnx := simplify(subs(r = 1, m[2] = N^2*m[1], rx),symbolic):
> rnx := subs(N = sqrt(n), rnx);
```

$$rnx := \frac{1}{\sqrt{n} + 1}$$

```
> plot(rnx, n=0..10, labels=['n', 'rx']);
```

图 22.2 表明平衡点的坐标为 m_2/m_1 的函数, 取 $r = 1$.

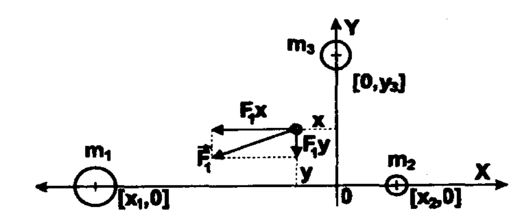
图 22.2 $\frac{r_x}{r} = f\left(\frac{m_1}{m_2}\right)$



22.3 三力的平衡

现在我们可以解三个质量分别为 m_1, m_2 和 m_3 的质点的类似问题. 它们的坐标标示在图 22.3 中. 设一个质点 m 在 (x, y) 处. 将由 m_i 产生的引力 $\vec{F_i}$ 表示成力 $F_i x$ 与 $F_i y$ 的和, 如图 22.3 所示.

图 22.3 三力平衡, 问题描述




```
> r1 := sqrt((x1 - x)^2 + y^2):
> r2 := sqrt((x2 - x)^2 + y^2):
> r3 := sqrt(x^2 + (y3 - y)^2):
> F1 := G*m1*m/r1^2:
> F2 := G*m2*m/r2^2:
> F3 := G*m3*m/r3^2:
> F1x := F1*(x1 - x)/r1:  F1y := -F1*y/r1:
> F2x := F2*(x2 - x)/r2:  F2y:= -F2*y/r2:
> F3x := -F3*x/r3:  F3y := F3*(y3 - y)/r3:
```

如果作用在这一点的合力为零, 质量 m 就集中在这个平衡点.

```
> Rx := F1x + F2x + F3x=0; Ry := F1y + F2y + F3y=0;
```

$$Rx := \frac{G\,m1\,m\,(x1 - x)}{(x1^2 - 2\,x1\,x + x^2 + y^2)^{3/2}} + \frac{G\,m2\,m\,(x2 - x)}{(x2^2 - 2\,x2\,x + x^2 + y^2)^{3/2}} - \frac{G\,m3\,m\,x}{(x^2 + y3^2 - 2\,y3\,y + y^2)^{3/2}} = 0$$

$$Ry := -\frac{G\,m1\,m\,y}{(x1^2-2\,x1\,x+x^2+y^2)^{3/2}} - \frac{G\,m2\,m\,y}{(x2^2-2\,x2\,x+x^2+y^2)^{3/2}} + \frac{G\,m3\,m\,(y3-y)}{(x^2+y3^2-2\,y3\,y+y^2)^{3/2}} = 0$$

我们得到一个关于 x 和 y 的非线性方程组，通常它没有解析解．对给定的值 $x_1=-5$, $x_2=3$, $y_3=8$, $m_1=12$, $m_2=9$ 和 $m_3=6$, 可用图像法求近似解．用 implicitplot(见图 22.4) 描出 $R_x(x,y)=0$ 和 $R_y(x,y)=0$, 我们可以确定解的存在区间．这些区间被 fsolve 用于数值计算．方程包含了变量 G 和 m 为公共因子，令 $G=1$ 和 $m=1$, 可把它们约去.

```
> System := subs(G = 1, m = 1, m1 = 12, x1 = -5, m2 = 9,
>                x2 = 3, m3 = 6, y3 = 8, {Rx, Ry});
```

$$System := \{-12\,\frac{y}{(25+10\,x+x^2+y^2)^{3/2}} - 9\,\frac{y}{(9-6\,x+x^2+y^2)^{3/2}} + 6\,\frac{8-y}{(x^2+64-16\,y+y^2)^{3/2}} = 0,$$
$$12\,\frac{-5-x}{(25+10\,x+x^2+y^2)^{3/2}} + 9\,\frac{3-x}{(9-6\,x+x^2+y^2)^{3/2}} - 6\,\frac{x}{(x^2+64-16\,y+y^2)^{3/2}} = 0\}$$

```
> with(plots):
> implicitplot(System, x = -4..2, y = 0..6, numpoints = 800,
>              labels = [`x`,`y`], color = black);
> Digits:=20:
> NS1 := fsolve(System, {x, y}, {x = -1..0, y = 0..1});
> NS2 := fsolve(System, {x, y}, {x = -1..0, y = 4..5});
```

$$NS1 := \{x = -.72487727980894034937, y = .30815197495485312203\}$$
$$NS2 := \{x = -.25295953560065612935, y = 4.3264261442756129585\}$$

```
> evalf(subs(NS1, System)); evalf(subs(NS2, System));
```

$$\{.42\,10^{-20} = 0, -.1\,10^{-19} = 0\}$$
$$\{-.2\,10^{-19} = 0, .1\,10^{-20} = 0\}$$

图 22.4 显示了两个相交点，故我们有两个解，两个平衡点．为数值计算我们取 20 位的精度，并用回代的方式验证解.

22.4 三力的平衡，由势能公式计算

与上一节同样问题可以通过找势能的临界点求解，因此我们要求满足 grad $U(x,y)=0$ 的坐标 x 和 y.

```
> Digits:=10:
> U := - G*m1/r1 - G*m2/r2 - G*m3/r3:
> with(linalg):
> g := grad(U,[x,y]);
```

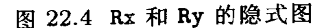
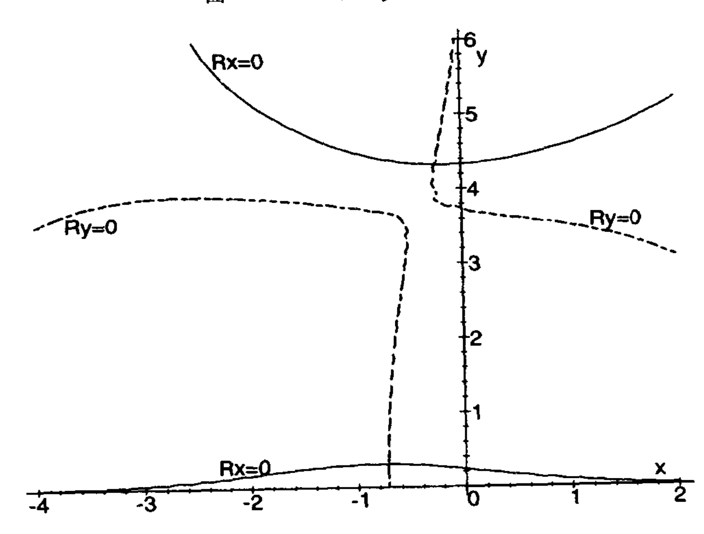


图 22.4 Rx 和 Ry 的隐式图



$$g := \left[\frac{1}{2}\,\frac{G\,m1\,(-2\,x1+2\,x)}{(x1^2-2\,x1\,x+x^2+y^2)^{3/2}}+\frac{1}{2}\,\frac{G\,m2\,(-2\,x2+2\,x)}{(x2^2-2\,x2\,x+x^2+y^2)^{3/2}}+\frac{G\,m3\,x}{(x^2+y3^2-2\,y3\,y+y^2)^{3/2}},\right.$$
$$\left.\frac{G\,m1\,y}{(x1^2-2\,x1\,x+x^2+y^2)^{3/2}}+\frac{G\,m2\,y}{(x2^2-2\,x2\,x+x^2+y^2)^{3/2}}+\frac{1}{2}\,\frac{G\,m3\,(-2\,y3+2\,y)}{(x^2+y3^2-2\,y3\,y+y^2)^{3/2}}\right]$$

当然这个解应该与前面得到的一样；此时，所得到方程组甚至与 22.3 节的一致．现在我们将绘出一条等势能曲线和一个 3 维的势能图．

```
> Un := subs(G = 1, m = 1, m1 = 12, x1 = -5,
>            m2 = 9, x2 = 3, m3 = 6, y3 = 8, U);
```

$$Un := -\frac{12}{\sqrt{25+10\,x+x^2+y^2}}-\frac{9}{\sqrt{9-6\,x+x^2+y^2}}-\frac{6}{\sqrt{x^2+64-16\,y+y^2}}$$

```
> gn := grad(Un, [x, y]);
```

$$gn := \left[6\,\frac{10+2\,x}{(25+10\,x+x^2+y^2)^{3/2}}+\frac{9}{2}\,\frac{-6+2\,x}{(9-6\,x+x^2+y^2)^{3/2}}+6\,\frac{x}{(x^2+64-16\,y+y^2)^{3/2}},\right.$$
$$\left.12\,\frac{y}{(25+10\,x+x^2+y^2)^{3/2}}+9\,\frac{y}{(9-6\,x+x^2+y^2)^{3/2}}+3\,\frac{-16+2\,y}{(x^2+64-16\,y+y^2)^{3/2}}\right]$$

```
> implicitplot({seq(Un = -i/6, i = 24..50)}, x=-6..3, y=-1..9,
>   scaling = constrained, labels = ['x', 'y'], color = black,
>   numpoints = 800);
> plot3d(Un, x = -6..3, y = -1..9, view = -12..-2,
>   orientation = [-100, 75], style = hidden, color = black,
>   numpoints = 40^2, axes = boxed, labels = ['x', 'y', 'U']);
```

我们现在重新计算能量的平衡点，并用 Hessian 确定我们所找到的临界点的类型．

图 22.5 等势能曲线

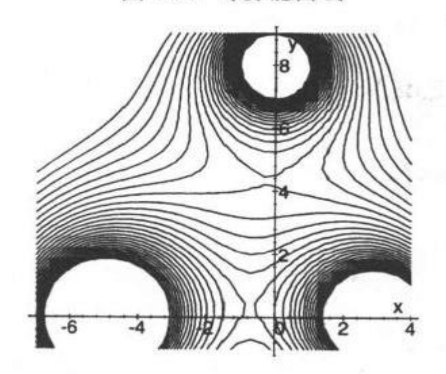
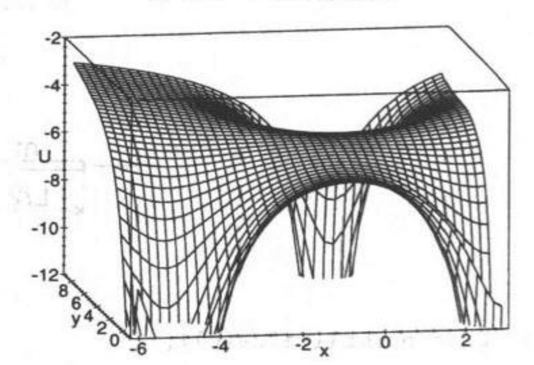


图 22.6 3 维的势能图.




```
> NS1 := fsolve({gn[1], gn[2]}, {x, y}, {x = -1..0, y= 0..1});
> NS2 := fsolve({gn[1], gn[2]}, {x, y}, {x = -1..0, y= 4..5});
```

$$NS1 := \{y = .3081519750, x = -.7248772798\}$$
$$NS2 := \{x = -.2529595356, y = 4.326426144\}$$

```
> subs(NS1, hessian(Un, [x, y]));
> subs(NS2, hessian(Un, [x, y]));
```

$$\begin{bmatrix} -.6309784820 & .006060066605 \\ .006060066605 & .2932060451 \end{bmatrix}$$
$$\begin{bmatrix} .08480382943 & -.01057859943 \\ -.01057859943 & -.3070140559 \end{bmatrix}$$

Hessian 矩阵对角线上元素的相反符号表明矩阵是不定的，两个平衡点是鞍点．当然这可以从图 22.5 和图 22.6 得到．通常由势能导出方程组的推导是简单的，且比用力分解的方法来得快．

22.5 粗线段的引力

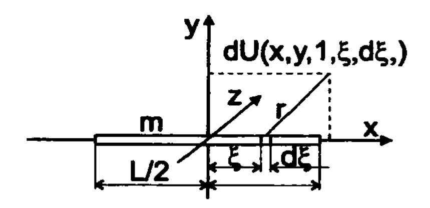
22.5.1 势能和强度

我们考虑求一条质量为 m, 长度为 l 的线段的引力场的引力势能和强度的问题.

```
> restart;
> with(plots):  with(linalg):
> r := sqrt((xi - x)^2 +y^2 +z^2):
> sigma := m/L: # linear mass density
> U := Int(G*sigma/r, xi = -L/2..L/2);
```

$$U := \int_{-1/2\,L}^{1/2\,L} \frac{G\,m}{L\,\sqrt{\xi^2 - 2\,\xi\,x + x^2 + y^2 + z^2}}\,d\xi$$

图 22.7 势能的计算




```
> U := normal(value(U));
```

$$U := -G\,m(-\ln(L - 2\,x + \sqrt{L^2 + 4\,x^2 + 4\,y^2 - 4\,L\,x + 4\,z^2})$$
$$+ \ln(-L - 2\,x + \sqrt{L^2 + 4\,x^2 + 4\,y^2 + 4\,L\,x + 4\,z^2}))\,/L$$

在与 xy 平面平行的平面 $z = 1/5$ 上我们绘出势能，等势曲线及强度向量场.

```
> Us := subs(G=1,m=1,L=1, z=1/5, U);
```

$$Us := \ln(1 - 2\,x + \sqrt{\frac{29}{25} + 4\,x^2 + 4\,y^2 - 4\,x}) - \ln(-1 - 2\,x + \sqrt{\frac{29}{25} + 4\,x^2 + 4\,y^2 + 4\,x})$$

```
> plot3d(Us, x = -2..2, y=-2..2, axes = boxed, style = hidden,
>          color = black, orientation = [-80, -130],
>          numpoints = 35^2, labels = ['x', 'y', 'U']);
> p1 := implicitplot({seq(Us = i/10, i = 5..30)}, x = -1..1,
>          y = -1..1, color = black, scaling = constrained):
> p2 := gradplot(Us, x = -1..1, y = -1..1, color = black,
>          scaling = constrained, arrows = SLIM, numpoints = 40^2):
> display({p1,p2},labels = ['x','y']);
```

图 22.9 可作为一个很好的例子说明向量场和等势曲线总是相互垂直的. 我们通过计算 $\lim_{L\to 0} U$ 和 $\lim_{L\to 0} g$, 来验证结论的正确性.

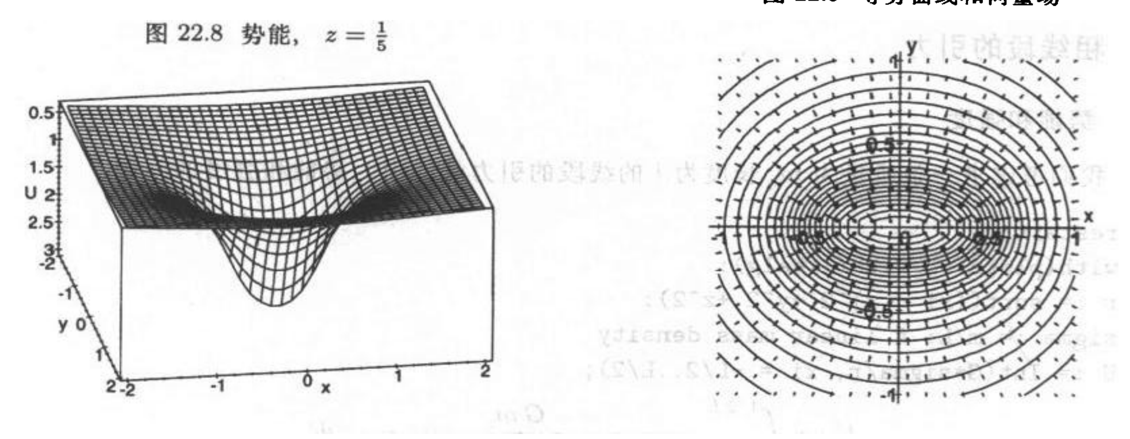


图 22.8 势能, $z = \frac{1}{5}$

图 22.9 等势曲线和向量场

```
> Uinf := Limit(U, L=0);
```

$$Uinf := \lim_{L \to 0} -G\,m(-\ln(L-2\,x+\sqrt{L^2+4\,x^2+4\,y^2-4\,L\,x+4\,z^2}) \\ +\ln(-L-2\,x+\sqrt{L^2+4\,x^2+4\,y^2+4\,L\,x+4\,z^2}))\,/L$$

```
> Uinf := value(Uinf);
```

$$Uinf := \frac{G\,m}{\sqrt{z^2+x^2+y^2}}$$

```
> ginf := grad(Uinf, [x, y, z]);
```

$$ginf := \left[-\frac{G\,m\,x}{(z^2+x^2+y^2)^{3/2}}, -\frac{G\,m\,y}{(z^2+x^2+y^2)^{3/2}}, -\frac{G\,m\,z}{(z^2+x^2+y^2)^{3/2}}\right]$$

```
> abs(ginf) = simplify(norm(ginf, 2), symbolic);
```

$$|ginf| = \frac{G\,m}{z^2+x^2+y^2}$$

对长距离, $r >> L$, 线段看起来象一个点, 引力场近似球对称. 对 $L \to 0$, U 和 g 的极限计算归结为关于一个点的经典 Newton 定律. 至此我们的结论是一致的.

22.5.2 质点的轨迹

我们将考察一个在粗线段的引力场移动的质点. 与此同时, 我们还将发现 Kepler 第二定律 — 速度场的等势面不适用于这种情况.

我们用 Newton 运动方程求数值解:

```
> Us := subs(G = 1, m = 1, L = 1, U):
> D2r := [diff(x(t), t, t), diff(y(t), t, t), diff(z(t), t, t)]:
> g := subs(x = x(t), y = y(t), z = z(t), grad(Us, [x, y, z])):
> IniC:= x(0) = 1, D(x)(0) = 0, y(0) = 0, D(y)(0) = 1,
>        z(0) = 3/4, D(z)(0) = 0:
> Ns := dsolve({seq(D2r[i] = g[i], i = 1..3), IniC},
>               {x(t), y(t), z(t)}, numeric);
```

$$Ns := \mathbf{proc}(rkf45_x)\ \ldots\ \mathbf{end}$$

不可能预先知道 MAPLE 以什么样的次序返回函数. 因此我们需要用算法 22.1 来确定变量在微分方程数值解中的次序. 算法 22.1 将函数的指数存于 C1, C2, C3, 将它们导数的指数存于 V1, V2, V3.

算法 22.1 dsnumsort 过程

```
dsnumsort := proc(numpr::list, Coor::list)
  local i, j, n;
  global C1, C2, C3, V1, V2, V3;
  n := nops(Coor):
  print(`Order of the variables:`);
  for i from 2 to 2*n + 1 do;
    for j from 1 to n do
      if [numpr[i]] =
```

```
                  select(has, numpr, diff(Coor[j](t),t)) then
              C.j := i - 1; V.j := i;
              print(Coor[j], C.j,'   ', diff(Coor[j](t),t), V.j);
            fi;
          od;
        od;
      end:
```

```
> read 'dsnumsort.map';
> dsnumsort(Ns(0), [x,y,z]):
```

$$Order\ of\ the\ variables:$$
$$y,\ 2,\quad ,\ \frac{\partial}{\partial t}\,\mathrm{y}(t),\ 3$$
$$x,\ 4,\quad ,\ \frac{\partial}{\partial t}\,\mathrm{x}(t),\ 5$$
$$z,\ 6,\quad ,\ \frac{\partial}{\partial t}\,\mathrm{z}(t),\ 7$$

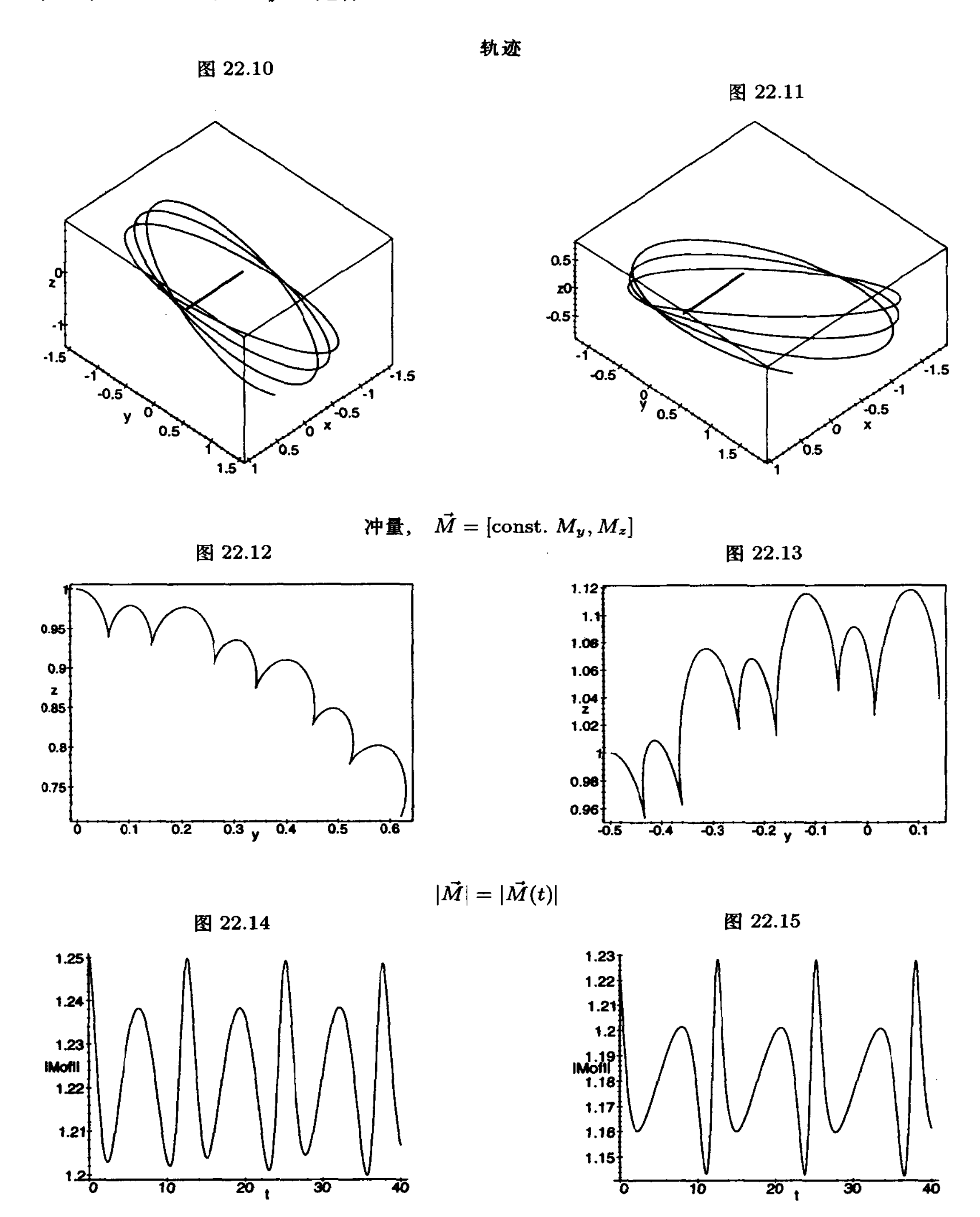
下面的 MAPLE 命令绘出了轨迹并验证了 Kepler 第二定律.

```
> for i from 0 to 1000 do;
>   T := i/25;
>   NsT := Ns(T):
>   X[i] := rhs(NsT[C1]); Vx[i] := rhs(NsT[V1]);
>   Y[i] := rhs(NsT[C2]); Vy[i] := rhs(NsT[V2]);
>   Z[i] := rhs(NsT[C3]); Vz[i] := rhs(NsT[V3]);
>   KepVec[i] := convert(crossprod([X[i], Y[i] ,Z[i]],
>     [Vx[i], Vy[i], Vz[i]]), list);
>   KepAbs[i] := norm(KepVec[i], 2);
> od:
> spacecurve({[seq([X[i], Y[i], Z[i]], i = 0..1000)],
>   [[-1/2, 0, 0], [1/2, 0, 0]]}, labels=['x', 'y', 'z']);
> spacecurve([seq(KepVec[i], i = 0..1000)],
>   orientation=[0,90], labels=['x', 'y', 'z']);
> plot([seq([i/25, KepAbs[i]], i = 0..1000)],
>   labels = ['t', 'MofI']);
```

对 40[时间单位] 和时间步长 = 1/25[时间单位], 结果在图 22.10,22.12 和 22.14 中给出. 图 22.11, 22.13 和 22.15 展示了质点的运动，初始条件为:

```
> IniC := x(0) = 1, D(x)(0) = 0, y(0) = 0, D(y)(0) = 1,
>         z(0) = 1/2, D(z)(0) = 1/2:
```

为了比较图形给出的数据，我们必须计算轨道的径向量和速度向量. 这些向量有助于我们计算 “表面速度”, 如果我们假设质点质量为 1, 它就是动量. 图 22.10 和图 22.11 中的轨迹不再是闭轨迹. Kepler 第二定律不成立，正如我们从图 22.12–22.15 看到的那样. 对一个球对称引力场，动量是常数. 此时由于沿 x 轴的旋转对称，仅 M_x 是常数. 当然在短距离内动量非常数，因为线段引力场没有球对称性，见图 22.9.



轨迹

图 22.10 图 22.11

冲量, $\vec{M} = [\text{const. } M_y, M_z]$

图 22.12 图 22.13

$|\vec{M}| = |\vec{M}(t)|$

图 22.14 图 22.15

22.6 人造地球卫星

现在我们将一个人造卫星放在绕地球的轨道上. 人造卫星在近地点 (最靠近地球的点) 的速度为 9000 m/s. 近地点的高度为 622 km. 我们求人造卫星的轨道并验证 Kepler 第二定律的有效性.

此时卫星的轨道是平面的，所以我们在平面上用 $x(t)$ 和 $y(t)$ 坐标解这个问题.

```
> restart;
> dx := diff(x(t), t, t) = -G*Mz*x(t)/(x(t)^2 + y(t)^2)^(3/2):
> dy := diff(y(t), t, t) = -G*Mz*y(t)/(x(t)^2 + y(t)^2)^(3/2):
> G:=6.67*10^(-11):  Mz:=6*10^24:
> Inic := x(0) = 7*10^(6), D(x)(0)=0, y(0)=0, D(y)(0)=9*10^3:
> Digits := 15:
> Ns := dsolve({dx, dy, Inic}, {x(t),y(t)}, numeric):
> read `dsnumsort.map`;
> dsnumsort(Ns(0), [x, y]);
```

$$\textit{Order of the variables}:$$
$$x, 2, \quad , \frac{\partial}{\partial t}\mathrm{x}(t), 3$$
$$y, 4, \quad , \frac{\partial}{\partial t}\mathrm{y}(t), 5$$

```
> for i from 0 to 400 do;
>   T := i*40;
>   NsT := Ns(T):
>   X[i] := rhs(NsT[C1]); Vx[i] := rhs(NsT[V1]);
>   Y[i] := rhs(NsT[C2]); Vy[i] := rhs(NsT[V2]);
>   MofI[i] := X[i]*Vy[i]-Y[i]*Vx[i];
> od:
> with(plots):
> p1 := polarplot(6378*10^3, phi = 0..2*Pi):
> p2 := plot([seq([X[i], Y[i]], i = 0..327)], thickness=2):
> display({p1, p2}, labels = ['x', 'y'], scaling = constrained);
> plot([seq([i*40,MofI[i] - 0.63*10^11], i = 0..400)],
>   labels=['t','Wp']);
```

图 22.16 轨迹，$1:10^3$ km

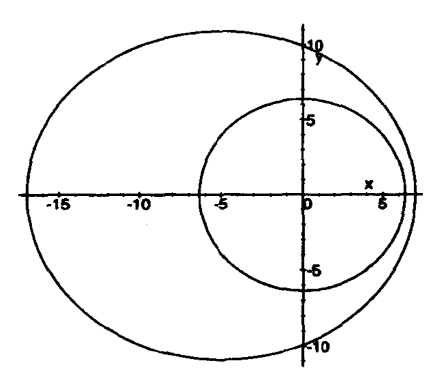
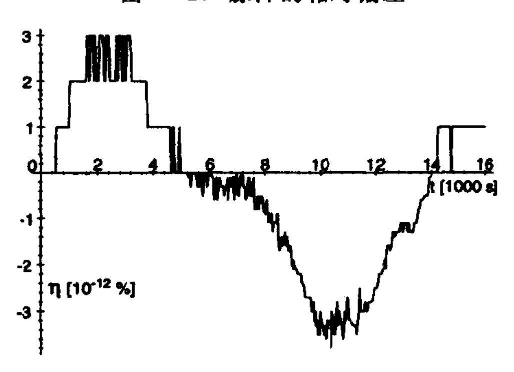


图 22.17 脉冲的相对偏差



MAPLE 的求解命令与前面的类似. Kepler 第二定律成立，正如我们可以由图 22.17 证实的那样，它显示了动量的相对偏差.

我们可以观察到，轨道的周期变化在第 326 和 327 时间段之间结束. 我们用二等分法确定轨道的周期.

```
> tx := evalf((t1+t2)/2):
> t1 := 326*40:  t2 := 327*40:
> y1 := rhs(Ns(t1)[C2]):
> y2 := rhs(Ns(t2)[C2]):
> while (t1 < tx) and (tx < t2) do:
>   yx := rhs(Ns(tx)[C2]):
>   if yx > 0 then
>     y2 := yx:  t2 := tx:
>    else
>     y1 := yx:  t1 := tx:
>   fi;
> od:
> Tx = evalf(tx);
```

$$Tx = 13060.2874343718$$

```
> Hx := floor(tx/3600):
> Mx := floor((tx - Hx*3600)/60):
> Sx := tx - Hx*3600 - Mx*60:
> Hx, Mx, Sx;
```

$$3, 37, 40.2874343718$$

卫星轨道周期是 3 小时 37 分 40.28746 秒.

我们保存主要的变量以便在后面几节中使用. 为此最好将加了索引的变量列出.

```
> XS := [seq(X[i], i = 0..328)]:  YS := [seq(Y[i], i = 0..328)]:
> VxS := [seq(Vx[i], i = 0..328)]:  VyS := [seq(Vy[i], i = 0..328)]:
> save(G, Mz, XS, YS, VxS, VyS,`orbit.sav`);
```

22.7 人造地球卫星, 第二解

问题: 运用在均匀引力场中的运动定律解例子 22.6. 假设在一个时间步长里人造卫星的位移很小, 使得引力加速度可以被假设为常数. 我们用著名的 Euler 方法对运动方程组积分.

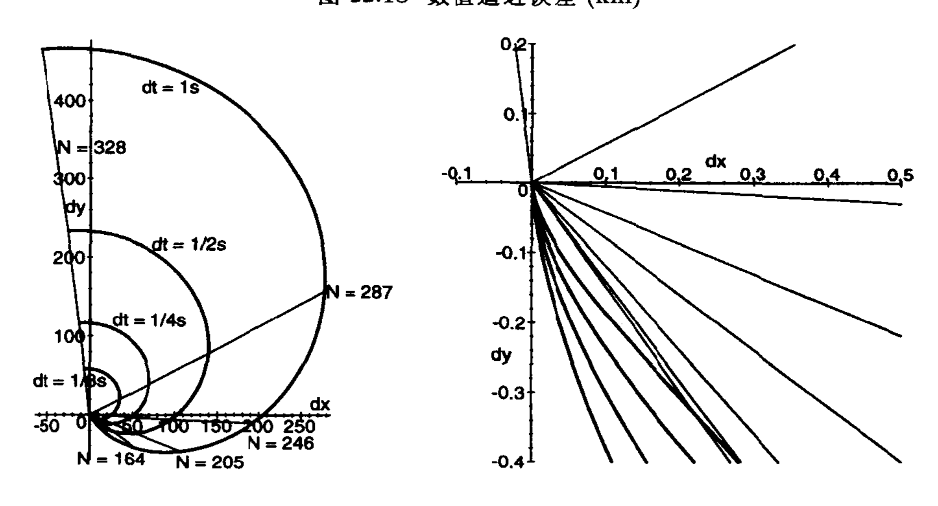
```
> restart;
> read(`orbit.sav`):  with(plots):
> ax := -G*Mz*x/(x^2 + y^2)^(3/2):
> ay := -G*Mz*y/(x^2 + y^2)^(3/2):
> i := 'i':  j := i + 1:
> for k from 0 to 3 do:
>   x := 7*10^6:  Vx := 0:
>   y := 0:  Vy := 9000:
>   dt := evalf(1/2^k);
>   for i from 0 to 328 do:
>      X[i] := evalf(x); Y[i] := evalf(y):
>      for n from 1 to 40*2^k do:
>         x := evalf(ax*dt^2/2 + Vx*dt + x);
>         y := evalf(ay*dt^2/2 + Vy*dt + y);
```

```
>         Vx := evalf(ax*dt + Vx);
>         Vy := evalf(ay*dt + Vy);
>     od:
>     if i mod 41 = 0 then
>        dX[k, i] := X[i] - XS[j]; dY[k, i] := Y[i] - YS[j];
>     fi;
>   od:
>   p[k] := plot([seq([(X[i] - XS[j])/1000,
>         (Y[i] - YS[j])/1000], i = 0..328)], color = black):
> od:
> p1 := display({seq(p[k], k = 0..3)}, thickness = 3):
> SI := [seq(i*41, i = 0..8)]:
> p2 := plot({seq([seq([dX[k, i]/1000, dY[k, i]/1000],
>        k = 0..1), [0, 0]], i = SI)}, color = black):
> display({p1, p2}, scaling = constrained, labels = ['dx', 'dy']);
> display({p1, p2}, view = [-0.1..0.5, -0.4..0.2],
>        scaling = constrained, labels = ['dx', 'dy']);
```

这个算法包含了系统离散化误差，可以通过选择较小的时间步长来减小这个误差．我们选择的时间步长从 dt=1 s 到 1/8 s.

图 22.18 表达了精确位置 (如在前一节中计算的) 与 N 个时间步长后近似位置之差 Δ. 如果我们将这些差连接起来，我们就看到螺线形的曲线，如果时间步长很小它也很小．研究图 22.18, 我们看到，当步长被二等分时，误差就减小 2 的因子．所以 $\Delta \sim C\mathrm{dt}$, 其中 C 近似是常数．对较小的步长需要较多的点．如果我们考察相应时间内的误差，就会注意到这些点位于直线上．而且，误差不是一致地增加，(见图 22.18); 对前 205 个时间步长，误差约为 130 km, 但在下面的 100 个步长中，误差增至大于 500 km.

图 22.18 数值逼近误差 (km)



22.8 丢失的螺钉

考虑一个太空人在宇宙飞船外修理东西．在时刻 $t=0$ 丢失一个螺钉．假设这个螺钉相对飞

船的速度是 1 m/s. 我们设飞船的初始速度与前一例相同，螺钉垂直于飞船轨道的相对速度为：$\vec{v} = (1, 0)$ m/s.

我们将在飞船的坐标系中计算螺钉的轨道和速度，即，飞船总是在坐标系的原点. 首先我们用从例题 22.7 平移的坐标系，保持它与星座相对方向.

```
> restart;
> read(`orbit.sav`):
> dx := diff(x(t), t, t) = -G*Mz*x(t)/(x(t)^2 + y(t)^2)^(3/2):
> dy := diff(y(t), t, t) = -G*Mz*y(t)/(x(t)^2 + y(t)^2)^(3/2):
> Inic := x(0) = 7*10^(6), D(x)(0)=1, y(0)=0, D(y)(0)=9*10^3:
> Ns := dsolve({dx, dy, Inic}, {x(t),y(t)}, numeric):
> read `dsnumsort.map`; dsnumsort(Ns(0), [x, y]);
```

Order of the variables :

$$y,\ 2,\quad,\ \frac{\partial}{\partial t}\,\mathrm{y}(t),\ 3$$

$$x,\ 4,\quad,\ \frac{\partial}{\partial t}\,\mathrm{x}(t),\ 5$$

```
> for i from 0 to 328 do;
>    T := i*40; NsT := Ns(T);
>    X[i] := rhs(NsT[C1]); Vx[i] := rhs(NsT[V1]);
>    Y[i] := rhs(NsT[C2]); Vy[i] := rhs(NsT[V2]);
> od:
> i := 'i':  j := i + 1:
> plot([seq([X[i] - XS[j], Y[i] - YS[j]], i = 0..328)],
>      labels = [`d x`, `d y`]);
> plot([seq([Vx[i] - VxS[j], Vy[i] - VyS[j]], i = 0..328)],
>      labels = [`d Vx`, `d Vy`]);
```

其次我们用一个旋转坐标系. 飞船仍在原点，但系统以这样的方式旋转使得 x 轴的负向总是指向地球球心. 太空人在上，在下，面对或背对飞船："在下" 意味着沿负 x 轴，"面对" 意味着沿 y 轴.

```
> R := sqrt(XS[j]^2 + YS[j]^2):
> Sin := YS[j]/R: Cos:=X[j]/R:
> GCX := (X[i] - XS[j])*Cos - (Y[i] - YS[j])*Sin:
> GCY := (X[i] - XS[j])*Sin + (Y[i] - YS[j])*Cos:
> GCVx := (Vx[i] - VxS[j])*Cos - (Vy[i] - VyS[j])*Sin:
> GCVy := (Vx[i] - VxS[j])*Sin + (Vy[i] - VyS[j])*Cos:
```

以 GC 开始的变量表示了在旋转坐标系中螺钉相对飞船的位置和速度向量. 这个系统象一个地心系统.

```
> plot([seq([GCX, GCY], i = 0..328)], labels = [`d x`, `d y`]);
> plot([seq([GCVx, GCVy], i = 0..327)], labels = [`d Vx`,`d Vy`]);
```

图 22.19 和 22.21 表明螺钉不会丢！太空人有机会抓到这个螺钉，只要他愿意等待一个轨道周期.

22.9 结论

1. MAPLE 极大地简化了物理中的数学应用. 考虑在这一章中用到的数学工具：符号运算，线性代数，方程的数值解，微分方程的数值解，数值积分和 MAPLE 图形. 在教学中对结果进行图形说明是非常重要的.
2. 因为在教室中 MAPLE 是解决问题的有力工具，所以重点可放在物理问题和用数学语言对它的描述上.
3. 恰当地使用 MAPLE, 不仅需要了解 MAPLE. 还必须具备很好的数学知识.
4. 经验：如果计算机代数系统被用作教学设施，许多现行的教学模式必须改变.

固定方向

图 22.19 螺钉位置

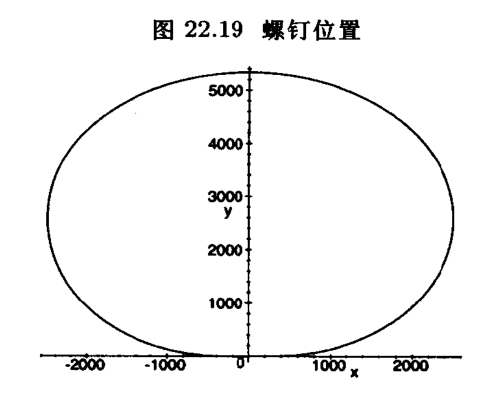
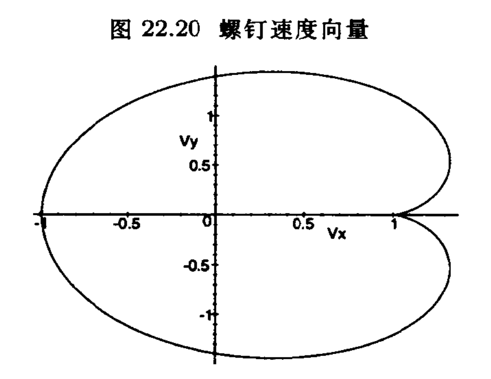


图 22.20 螺钉速度向量



地球定向系统

图 22.21 螺钉位置

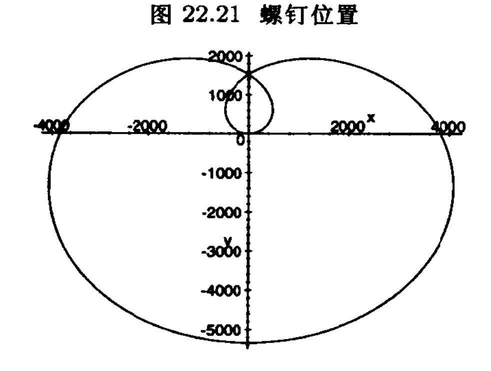
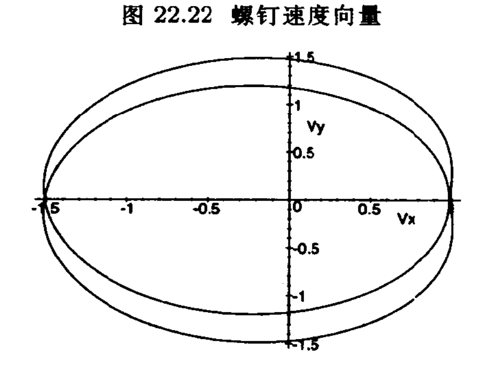


图 22.22 螺钉速度向量



参考文献

[1] JOURNAL OF CELESTIAL MECHANICS: *J. Henrard* editor, Belbium
[2] W. GANDER, *Computer–Mathematik*. Birkhauser, 1992
[3] E. GETTYS, F. KELLER, M. SKOVE, *Classical and Modern Physics*, McGrawHill, 1989
[4] H. YOUNG, R. FREEDMAN, *University Physics*, Addison Wesley, 1996

第二十三章　点云的最小二乘拟合

W. Gander

23.1　引言

我们考虑 坐标度量衡学 中 (参见 [2],[1]) 的一个最小二乘问题：当一个工作板是在参考框架中的 标称位置 时，它上面的 m 个点，所谓 标称点，由它们在构造平面的精确坐标给定，我们用

$$\mathbf{x}_1,\ldots,\mathbf{x}_m,\quad \mathbf{x}_i\in R^n,\ 1\le n\le 3$$

表示标称点在这个位置的坐标向量. 现在假设一个坐标度量仪采集到与另一个工作板相同的点. 这个仪器记录 测量点 的坐标

$$\xi_1,\ldots,\xi_m,\quad \xi_i\in R^n,\ 1\le n\le 3$$

它们是在一个不同于参考框架的框架中. 我们要解决的问题是求出从给定的标称点映射到测量点的坐标变换.

我们需要求一个平移向量 $\mathbf{t}$ 和一个 $\det(Q)=1$ 的正交矩阵 Q, 即 $Q^{\mathrm{T}}Q=I$, 使得

$$\xi_i=Q\mathbf{x}_i+\mathbf{t},\quad i=1,\ldots,m. \tag{23.1}$$

在三维空间中 $m>6$, 方程 (23.1) 是一个超定方程组，只有测量没有误差，它才是一致的. 对于实际的仪器，这是不可能的. 因此我们的问题是，确定最小二乘问题

$$\xi_i\approx Q\mathbf{x}_i+\mathbf{t} \tag{23.2}$$

的未知数 Q 和 $\mathbf{t}$. Hanson 和 Norris[4] 已经很好地研究并解决了这个问题.

23.2　计算平移

在一维的情况，给定直线上两个点集. 矩阵 Q 正好是常数 1, 我们必须确定一个数量 t, 使得

$$\xi_i\approx x_i+t,\quad i=1,\ldots,m.$$

令 $A=(1,\ldots,1)^{\mathrm{T}},\mathbf{a}=(\xi_1,\ldots,\xi_m)^{\mathrm{T}}$ 和 $\mathbf{b}=(x_1,\ldots,x_m)^{\mathrm{T}}$, 此问题变成

$$At\approx\mathbf{a}-\mathbf{b}. \tag{23.3}$$

根据法方程 $A^{\mathrm{T}}At=A^{\mathrm{T}}(\mathbf{a}-\mathbf{b})$, 我们得到 $mt=\sum_{i=1}^m(\xi_i-x_i)$, 因此

$$t=\bar{\xi}-\bar{x},\ 其中\ \ \bar{\xi}=\frac{1}{m}\sum_{i=1}^m\xi_i,\quad \bar{x}=\frac{1}{m}\sum_{i=1}^m x_i. \tag{23.4}$$

对于 $n>1$, 我们能推广此结果. 考虑

$$\xi_i\approx\mathbf{x}_i+\mathbf{t},\quad i=1,\ldots,m.$$

用矩阵符号，这个最小二乘问题变成 (I 是 $n \times n$ 的单位矩阵)：

$$\begin{pmatrix} I \\ I \\ \vdots \\ I \end{pmatrix} \mathbf{t} \approx \begin{pmatrix} \xi_1 - \mathbf{x}_1 \\ \xi_2 - \mathbf{x}_2 \\ \vdots \\ \xi_m - \mathbf{x}_m \end{pmatrix}$$

法方程是 $m\mathbf{t} = \sum_{i=1}^{m}(\xi_i - \mathbf{x}_i)$, 于是

$$\mathbf{t} = \bar{\xi} - \bar{\mathbf{x}},\ \text{其中}\ \bar{\xi} = \frac{1}{m}\sum_{i=1}^{m}\xi_i,\ \text{和}\ \bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i.$$

因此，我们已经证明， 平移 $\mathbf{t}$ 是连接对应点集的两个重心的向量.

23.3 计算正交矩阵

将前一节的结果，应用于最小二乘问题

$$\xi_i \approx Q\mathbf{x}_i + \mathbf{t} \Longleftrightarrow \sum_{i=1}^{m} \|Q\mathbf{x}_i + \mathbf{t} - \xi_i\|^2 = \min \tag{23.5}$$

得到 $\mathbf{t}$ 是连接点集 ξ_i 和 $Q\mathbf{x}_i$ 的两个重心的向量，即，

$$\mathbf{t} = \bar{\xi} - Q\bar{\mathbf{x}}. \tag{23.6}$$

利用方程 (23.6), 在 (23.5) 中消去 $\mathbf{t}$, 于是问题变成

$$G(Q) = \sum_{i=1}^{m} \|Q(\mathbf{x}_i - \bar{\mathbf{x}}) - (\xi_i - \bar{\xi})\|^2 = \min. \tag{23.7}$$

引进新的坐标

$$\mathbf{a}_i := \mathbf{x}_i - \bar{\mathbf{x}}\ \text{和}\ \mathbf{b}_i := \xi_i - \bar{\xi}$$

问题变为：

$$G(Q) = \sum_{i=1}^{m} \|Q\mathbf{a}_i - \mathbf{b}_i\|^2 = \min. \tag{23.8}$$

我们用向量组成矩阵

$$A := (\mathbf{a}_1, \ldots, \mathbf{a}_m)\ \text{和}\ B := (\mathbf{b}_1, \ldots, \mathbf{b}_m),$$

并利用 Frobenius 范数重写函数 G

$$G(Q) = \|QA - B\|_F^2.$$

因为一个矩阵和它的转置矩阵的 Frobenius 范数是相同的，所以我们最后得到一个 Procrustes 问题 [5]: 求一个正交矩阵 Q, 使得

$$\|B^{\mathrm{T}} - A^{\mathrm{T}}Q^{\mathrm{T}}\|_F^2 = \min. \tag{23.9}$$

23.4 Procrustes 问题的解法

我们考虑一个问题：给定两个 $m \times n$ 矩阵 C 和 D, 其中 $m \geq n$, 求一个 $n \times n$ 正交矩阵 P, 使得

$$\|C - DP\|_F^2 = \min.$$

我们需要 Frobenius 范数的一些性质．其定义是

$$\|A\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{i,j}^2 = \sum_{j=1}^{n} \|\mathbf{a}_j\|_2^2, \text{ 其中 } \mathbf{a}_j \text{ 是 } A \text{ 的第 } j \text{ 列}. \tag{23.10}$$

注意

$$\|A\|_F^2 = \operatorname{trace}(A^{\mathrm{T}}A) = \sum_{i=1}^{m} \lambda_i(A^{\mathrm{T}}A). \tag{23.11}$$

记住矩阵的迹是对角线元素之和，并且它等于特征值之和．方程 (23.11) 给出一些有用的关系：

1. 如果 P 是正交的，则 $\|PA\|_F = \|A\|_F$. 另外，因为 $\|A\|_F = \|A^{\mathrm{T}}\|_F$, 所以有 $\|AP\|_F = \|A\|_F$.
2.
$$\begin{aligned}
\|A+B\|_F^2 &= \operatorname{trace}((A+B)^{\mathrm{T}}(A+B)) \\
&= \operatorname{trace}(A^{\mathrm{T}}A + B^{\mathrm{T}}A + A^{\mathrm{T}}B + B^{\mathrm{T}}B) \\
&= \operatorname{trace}(A^{\mathrm{T}}A) + 2\operatorname{trace}(A^{\mathrm{T}}B) + \operatorname{trace}(B^{\mathrm{T}}B) \\
\|A+B\|_F^2 &= \|A\|_F^2 + \|B\|_F^2 + 2\operatorname{trace}(A^{\mathrm{T}}B).
\end{aligned}$$

现在我们将最后的关系式应用到 Procrustes 问题：

$$\|C - DP\|_F^2 = \|C\|_F^2 + \|D\|_F^2 - 2\operatorname{trace}(P^{\mathrm{T}}D^{\mathrm{T}}C) = \min.$$

计算最小值等价于最大化

$$\operatorname{trace}(P^{\mathrm{T}}D^{\mathrm{T}}C) = \max.$$

利用 $D^{\mathrm{T}}C$ 的奇异值分解 $U\Sigma V^{\mathrm{T}}$, 可得到

$$\operatorname{trace}(P^{\mathrm{T}}D^{\mathrm{T}}C) = \operatorname{trace}(P^{\mathrm{T}}U\Sigma V^{\mathrm{T}}).$$

因为 U 和 V 是正交的，所以我们可以把未知矩阵 P 写成如下形式

$$P = UZ^{\mathrm{T}}V^{\mathrm{T}}, \text{ 其中 } Z \text{ 是正交的}.$$

于是

$$\begin{aligned}
\operatorname{trace}(P^{\mathrm{T}}D^{\mathrm{T}}C) &= \operatorname{trace}(VZU^{\mathrm{T}}U\Sigma V^{\mathrm{T}}) = \operatorname{trace}(VZ\Sigma V^{\mathrm{T}}) \\
&= \operatorname{trace}(Z\Sigma VV^{\mathrm{T}}) = \operatorname{trace}(Z\Sigma) \\
&= \textstyle\sum_{i=1}^{n} z_{ii}\sigma_i \leq \sum_{i=1}^{n} \sigma_i,
\end{aligned}$$

其中对于任意正交矩阵 Z, 由 $z_{ii} \leq 1$ 可得上面不等式．而且当 $Z = I$ 时，可达到边界．注意，如果 $D^{\mathrm{T}}C$ 不是满秩，则解不唯一 (参见 [4])．从而我们证明了下列定理：

定理 *利用 $D^{\mathrm{T}}C$ 的正交极因子，即 $P = UV^{\mathrm{T}}$, 其中 $U\Sigma V^{\mathrm{T}}$ 是 $D^{\mathrm{T}}C$ 的奇异值分解，可求解 Procrustes 问题 $\|C - DP\|_F^2 = \min$.*

矩阵的极分解是复数的极表示的推广. 矩阵分解成一个正交矩阵乘以一个对称 (半) 正定矩阵的积. 利用奇异值分解或其它算法 [3], 可计算此分解. 在我们这里, 有

$$D^{\mathrm{T}}C = U\Sigma V^{\mathrm{T}} = \underbrace{UV^{\mathrm{T}}}_{\text{正交}} \quad \underbrace{V\Sigma V^{\mathrm{T}}}_{\text{半正定}}.$$

23.5 算法

给定测量点 ξ_i 和对应的标称点 $\mathbf{x}_i, i = 1, \ldots, m$. 我们要确定 $\mathbf{t}$ 和正交的 Q, 使得 $\xi_i \approx Q\mathbf{x}_i + \mathbf{t}$.

1. 计算重心:
$$\bar{\xi} = \frac{1}{m}\sum_{i=1}^{m}\xi_i \text{ 和 } \bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i.$$
2. 计算相关的坐标:
$$A := (\mathbf{a}_1, \ldots, \mathbf{a}_m), \qquad \mathbf{a}_i := \mathbf{x}_i - \bar{\mathbf{x}}$$
$$B := (\mathbf{b}_1, \ldots, \mathbf{b}_m), \qquad \mathbf{b}_i := \xi_i - \bar{\xi}$$
3. 求解 Procrustes 问题 $\|C - DP\|_F^2 = \min$, 其中 $C = B^{\mathrm{T}}, D = A^{\mathrm{T}}$ 和 $P = Q^{\mathrm{T}}$. 首先计算奇异值分解
$$AB^{\mathrm{T}} = U\Sigma V^{\mathrm{T}}.$$
4. $Q^{\mathrm{T}} = UV^{\mathrm{T}}$ 或 $Q = VU^{\mathrm{T}}$.
5. $\mathbf{t} = \bar{\xi} - Q\bar{\mathbf{x}}$.

因为技术原因, 把正交矩阵 Q 分解成初等旋转至关重要. 迄今为止, 我们给出的算法能计算一个正交矩阵, 但是不能保证, Q 表示成一些旋转的乘积, 并且没有反射发生. 如果 $\det(Q) = 1$, 则 Q 可表示成一些旋转的乘积. 然而, 如果 $\det(Q) = -1$, 则必定会有一个反射, 这可能没有实际用途. 因此我们应求 $\det(Q) = 1$ 的最好的正交矩阵.

在 [4] 中已证明, 约束的 Procrustes 问题

$$\|C - DP\|_F^2 = \min, \quad \text{服从于 } \det(P) = 1$$

其解为

$$P = U\mathrm{diag}(1, \ldots, 1, \mu)V^{\mathrm{T}},$$

其中 $D^{\mathrm{T}}C = U\Sigma V^{\mathrm{T}}$ 是奇异值分解, $\mu = \det(UV^{\mathrm{T}})$.

它的证明是基于这样一个事实, 对于一个 $n \times n$ 的实正交矩阵 Z, 其中 $\det(Z) < 1$, 它的迹有上界

$$\mathrm{trace}(Z) \leq n - 2, \text{ 并且 } \mathrm{trace}(Z) = n - 2 \Longleftrightarrow \lambda_i(Z) = \{1, \ldots, 1, -1\}.$$

这可由 Z 的实 Schur 型 [5] 证明. 因此 $\mathrm{trace}(Z\Sigma)$ 的最大值是 $\sum_{i=1}^{n-1}\sigma_i - \sigma_n$, 并在 $Z = \mathrm{diag}(1, \ldots, 1, -1)$ 时达到. 于是我们得到算法 23.1 所示的 MATLAB 函数 `pointfit`.

算法 23.1

```
function [t, Q] = pointfit(xi,x);
%
xiq = sum(xi')/length(xi); xiq = xiq';
```

```
xq = sum(x')/length(x); xq = xq';
A = x - xq*ones(1,length(x));
B = xi - xiq*ones(1,length(xi));
[u sigma v] = svd(A*B');
Q = v*diag([ones(1,size(v,1)-1) det(v*u')])*u';
t = xiq - Q*xq;
```

23.6 分解正交矩阵

正如前面提到的，正交矩阵 Q 分解成初等旋转是有用的. 在 [1] 中，从给定的标称点映射到测量点的仿射变换被写成

$$\xi = R_3R_2R_1U_0(\mathbf{x}-\mathbf{x_0})$$

其中

$$R_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_1 & s_1 \\ 0 & -s_1 & c_1 \end{pmatrix}, R_2 = \begin{pmatrix} c_2 & 0 & s_2 \\ 0 & 1 & 0 \\ -s_2 & 0 & c_2 \end{pmatrix} \text{ 和 } R_3 = \begin{pmatrix} c_3 & s_3 & 0 \\ -s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{23.12}$$

是平面旋转矩阵，它们分别由 $c_k = \cos\theta_k$ 和 $s_k = \sin\theta_k, k = 1,2,3$, 定义关于 x,y 和 z 轴的旋转，U_0 是一个给定不变的正交矩阵.

从已知变换 $\xi = Q\mathbf{x}+\mathbf{t}$, 我们容易计算矩阵 R_k 和向量 $\mathbf{x_0}$. 向量 $\mathbf{x_0}$ 与 $\mathbf{t}$ 的关系是 $-Q\mathbf{x_0} = \mathbf{t}$, 于是

$$\mathbf{x_0} = -Q^{\mathrm{T}}\mathbf{t}.$$

为了计算矩阵 R_k, 首先我们注意

$$Q = R_3R_2R_1U_0 \Longleftrightarrow R_1^{\mathrm{T}}R_2^{\mathrm{T}}R_3^{\mathrm{T}}QU_0^{\mathrm{T}} = I.$$

从而我们可进行下列步骤:

1. 构成 $H = QU_0^{\mathrm{T}}$.
2. 确定旋转角 θ_3, 使得在 $H := R_3^{\mathrm{T}}H$ 中，元素 h_{21} 变成 0:

$$\begin{pmatrix} c_3 & -s_3 & 0 \\ s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{pmatrix} H = \begin{pmatrix} * & * & * \\ 0 & * & * \\ * & * & * \end{pmatrix} \Longleftrightarrow \begin{aligned} \tan\theta_3 &= -\tfrac{h_{21}}{h_{11}}, \\ c_3 &= \cos\theta_3, \\ s_3 &= \sin\theta_3. \end{aligned}$$

3. 类似地，在 $H := R_2^{\mathrm{T}}H$ 中消去 h_{31}:

$$\tan\theta_2 = -\frac{h_{31}}{h_{11}},\ c_2 = \cos\theta_2,\ s_2 = \sin\theta_2.$$

4. 最后，在 $H := R_1^{\mathrm{T}}H$ 中，旋转 h_{32} 为 0:

$$\tan\theta_1 = -\frac{h_{32}}{h_{22}},\ c_1 = \cos\theta_1,\ s_1 = \sin\theta_1.$$

算法 23.2

```
function [theta] = rotangle(H)
%
if det(H)<0
  error('The matrix is not a product of rotations')
end
n = size(H,1);
theta = [];
for i = 1 : n - 1
  for j = i + 1 : n
    theta_k = atan2(-H(j,i),H(i,i));
    theta = [theta_k, theta];
    c = cos(theta_k); s = sin(theta_k);
    R = eye(n);
    R(i,i) = c; R(j,j) = c;
    R(i,j) = -s; R(j,i) = s;
    H = R * H;
  end
end
```

因而我们得到算法 23.2 所示的 MATLAB 函数 rotangle.

23.7 数值例子

我们以两个三维的例子和一个测试函数 pointfit 和 rotangle 的程序结束本章.

23.7.1 第一个例子

首先我们定义一个棱锥的节点

```
>> A = [ 1  0  0  0;
>>       0  2  0  0;
>>       0  0  3  0 ];
```

为了画出此棱锥，必须定义一个向量，它表示所绘制的边.

```
>> v = [ 1   2   3   4   1   3   4   2];
>> plot3(A(1,v), A(2,v), A(3,v), '-.');
>> hold on;
>> view(115,20);
>> axis([-2 2 0 5 0 4]);
```

然后，在此棱锥上选取一些标称点

```
>> x = [ 0   0.5  0.5  0     0     0    1;
>>       0   0    1    1.5   0.5   0    0;
```

```
>>         0   1.5  0    0.75  2.25  2    0 ];
>> plot3(x(1,:), x(2,:), x(3,:), '*');
```

算法 23.3

```
function [Qr] = ang2orth(theta)
% generate orthogonal matrix from given angles
Qr = eye(3);
n = 1;
for i = 2 : -1 : 1
  for j = 3 : -1 : i + 1
    t = thetar(n);
    s = sin(t); c = cos(t);
    U = eye(3);
    U(i,i) =  c; U(i,j) = s;
    U(j,i) = -s; U(j,j) = c;
    Qr = U * Qr; n = n + 1;
  end
end
```

现在我们将模拟测量点. 为了给读者可再产生的结果, 我们注释出随机产生的数据, 用固定的数值代替它们. 通过产生三个角和利用算法 23.3 所示的 MATLAB 函数 ang2orth, 我们构造一个正交矩阵 Qr:

```
>> %thetar = rand(1,1:3);
>> thetar = [pi/4 pi/15 -pi/6];
>> Qr = ang2orth(thetar);
```

和一个随机平移向量:

```
>> %tr = rand(3,1)*3;
>> tr = [1;3;2];
```

现在我们能计算, 并画出这个精确变换的棱锥:

```
>> B = Qr * A + tr * ones(1, size(A,2));
>> plot3(B(1,v), B(2,v), B(3,v), ':');
>> pause
```

我们变换标称点, 并加上某噪声以模拟测量点 (又因为结果的再产生性, 我们明确地给出这些点):

```
>> % (randomly distorted) measured points;
>> % xi = (Qr*x+tr*ones(1,length(x))+randn(size(x))/10);
>>
>> xi = [ 0.8314 0.9821 1.0211 0.1425 0.2572 0.5229 1.7713
>>        3.0358 4.5232 3.8075 4.4826 5.0120 4.5364 3.3987
>>        1.9328 2.8703 1.0573 1.5803 3.1471 3.5394 1.9054];
>> plot3(xi(1,:), xi(2,:), xi(3,:), 'o');
>> pause
```

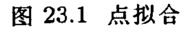
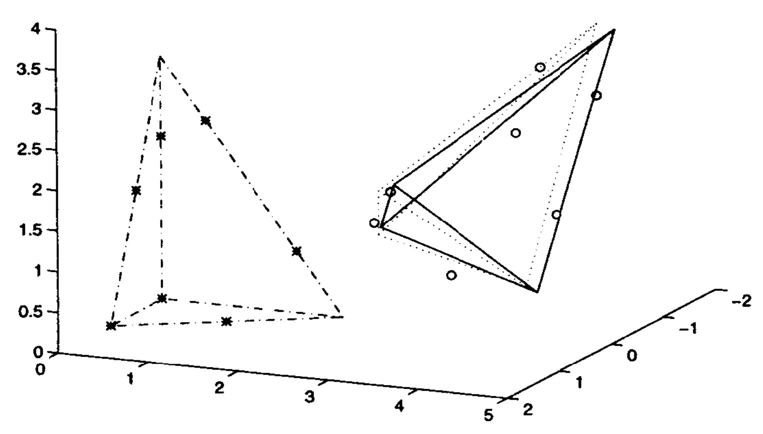


图 23.1 点拟合



利用函数 **pointfit**, 我们由测量点估计 Q 和 t, 并画出这个拟合的棱锥 (参见图 23.1):

```
>> [t,Q] = pointfit(xi,x);
>> % the fitted pyramid:
>> C = Q * A + t * ones(1, size(A,2));
>> plot3(C(1,v), C(2,v), C(3,v), '-');
>> hold off;
```

最后我们计算这个拟合棱锥的旋转角，并与原来的进行比较.

```
>> theta = rotangle(Q)
theta =
    0.8282    0.1772   -0.3964
>> thetar
thetar =
    0.7854    0.2094   -0.5236
```

作为最后的核对，我们由计算的角 θ 产生正交矩阵 S, 并与 `pointfit` 的结果 Q 比较:

```
>> S = ang2orth(theta)>;
>> norm(Q - S)
ans =
    2.3497e-015
```

23.7.2 第二个例子

在这个例子中，我们构造两个点集，它们可由一个反射精确地相互变换. 因为物体的方向会改变，旋转与测量点不匹配. 我们将只利用这些旋转，计算出最好的解. 再把棱锥的顶点看作标称点:

```
>> % Define nominal points.
>> A = [1 0 0 0;0 2 0 0;3 3 3 0];
>> v = [1 2 3 4 1 3 2 4];
```

```
>> plot3(A(1,v), A(2,v), A(3,v), '-.');
>> hold on;
>> view(110,20)
>> axis([-1 2 -1 3 -3 3])
```

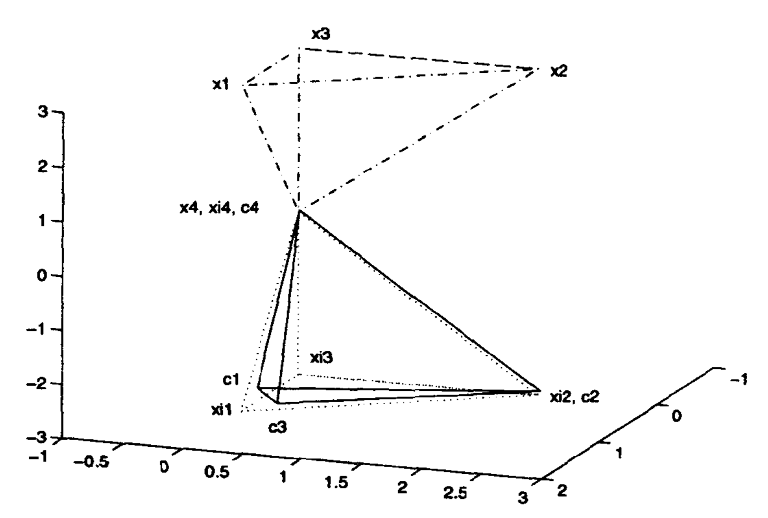
我们用棱锥的顶点作为测量点，这些顶点在 xy 平面反射：

```
>> B = [1 0 0 0;0 2 0 0;-3 -3 -3 0];
>> plot3(B(1,v), B(2,v), B(3,v), ':')
```

我们计算最好的拟合，并在图 23.2 画出结果：

```
>> [t,Q] = pointfit(B,A);
>> C = Q * A + t * ones(1, size(A,2))
>> plot3(C(1,v), C(2,v), C(3,v), '-')
>> text(1,-.25,3,'x1'); text(0,2.1,3,'x2')
>> text(0,.1,3.3,'x3'); text(0,-1,-0.1,'x4, xi4, c4')
>> text(1,-.25,-3,'xi1'); text(0,2.1,-3,'xi2, c2')
>> text(0,.1,-2.65,'xi3'); text(0.7,-0.3,-2.65,'c1')
>> text(1.75,.6,-2.65,'c3')
>> theta = 180/pi*rotangle(Q)
>> hold off;
```

图 23.2 最佳旋转：x_i 标称点，xi_i"测量点"，c_i 最佳拟合



参考文献

[1] B. P. Butler, A. B. Forbes and P. M. Harris, *Algorithms for Geometric Tolerance Assessment*, Technical Report DITC 228/94, National Physical Laboratory, Teddington, 1994.

[2] A. B. Forbes, *Geometric Tolerance Assessment*, Technical Report DITC 210/92, National Physical Laboratory, Teddington, 1992.

[3] W. GANDER, *Algorithms for the Polar Decomposition*, SIAM J. on Sci. and Stat. Comp., 11:6(1990).

[4] R. HANSON AND M. NORRIS, *Analysis of Measurements Based on the Singular Value Decomposition*, SIAM J. on Sci. and Stat. Comp., 2:3(1981).

[5] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 2nd ed, Baltimore Johns Hopkins University Press, 1989.

第二十四章　社会过程建模

J. Hřebíček and T. Pitner

24.1　引言

某些社会过程可以使用由 Weidlich 和 Haag 介绍的模型来研究 (参见 [3],[5]). 他们企图开发一个定量社会学的一般背景. 这一章我们将应用他们的方法使用 MAPLE 来求解两个模型.

社会系统演化建模的一般方法基于一个特征空间, 它被定义为一个线性向量空间. 它的向量根据社会成员在社会中的行为特征把他们描述为一个社会群体 [6]. 这些特征可以是职业, 偏向于某个政党, 生活标准, 收入水平等.

特征空间的向量根据已知特征 α 把系统中的一个成员归入某个群体 i. 令 m_i^α 表示这些成员的个数. 社会系统的演化是由改变群体成员对特征 α 的看法来确定的.

我们定义一个转移概率 $w_{ij}^\alpha(t)$, 表示单位时间由特征 α 表示的成员的看法发生变化, 也就是从群体 j 转入群体 i 的概率. 则 m_i^α 的增量将由如下的主方程给出

$$\frac{\mathrm{d}m_i^\alpha(t)}{\mathrm{d}t}=\sum_{\substack{j=1\\ j\neq i}} m_j^\alpha(t)w_{ij}^\alpha(t)-m_i^\alpha(t)w_{ji}^\alpha(t), \tag{24.1}$$

它是描述一类社会系统的基本方程 [5]. 这个主方程可以描述具有空间结构 (如人口迁移) 和时间震荡 (如 Schumpeter 钟 [4]) 的过程.

24.2　人口迁移建模

方程 (24.1) 的一个有趣的应用是建立生活在两个城市居民区 ($\alpha=2$) 的两群居民 ($s=2$) 和他们的移民之间的关系模型. 这两群人分别用 m 和 n 来表示. 它是来源于 Atlanda(Georgia,USA) 的白人和黑人居民的移民模型 (参见 [6]).

假设两群居民的人数都是常数同时分别等于 $2m_0$ 和 $2n_0$. 这两群人在时间 t 散布在两个城市的居民区中. 在每个居民区的人数分别是 $m_1(t),m_2(t)$ 和 $n_1(t),n_2(t)$. 则 $m_1(t)+m_2(t)=2m_0$ 和 $n_1(t)+n_2(t)=2n_0$.

引入新的变量 $m(t)=m_1(t)-m_0=m_0-m_2(t)$ 和 $n(t)=n_1(t)-n_0=n_0-n_2(t)$, 我们能够把主方程 (24.1) 化简为每群人一个方程. 我们得到如下的方程组 (它们在当前是没有关系的)

$$\begin{aligned}\frac{\mathrm{d}m(t)}{\mathrm{d}t} &= (m_0-m(t))w_{12}^m(t)-(m_0+m(t))w_{21}^m(t),\\ \frac{\mathrm{d}n(t)}{\mathrm{d}t} &= (n_0-n(t))w_{12}^n(t)-(n_0+n(t))w_{21}^n(t),\end{aligned} \tag{24.2}$$

现在我们需要导出转移概率函数 $w_{ij}^\alpha(t)$ 的模型. 对于每个人群 m,n 要用到三个常数:

1. 自然优先参数 a_m,a_n: 描述人群 m,n 生活在同一个城市居民区的自然趋势,
2. 内部倾向参数 b_m,b_n: 描述人群 m,n 与同一群人生活在同一个城市居民区的趋势,

3. 外部倾向参数 c_m, c_n: 描述人群 m, n 与另一群人生活在同一个城市居民区的趋势.
则转移概率函数可以由使用这些参数的指数函数来定义 (参见 [6]):

$$\begin{aligned} w_{12}^m(t) &= Ae^{a_m+b_m m(t)+c_m n(t)} \\ w_{21}^m(t) &= Ae^{-(a_m+b_m m(t)+c_m n(t))} \\ w_{12}^n(t) &= Ae^{a_n+b_n m(t)+c_n n(t)} \\ w_{21}^n(t) &= Ae^{-(a_n+b_n m(t)+c_n n(t))} \end{aligned} \tag{24.3}$$

其中 A 是时间尺度因子.

引入新的变量 $x(t) = m(t)/m_0, y(t0 = n(t)/n_0$, 同时表示

$$\begin{aligned} u(t) &= a_m + b_m m_0 x(t) + c_m n_0 y(t), \\ v(t) &= a_n + b_n m_0 x(t) + c_n n_0 y(t), \end{aligned}$$

从 (24.2) 和 (24.3) 我们得到

$$\begin{aligned} \frac{\mathrm{d}x(t)}{\mathrm{d}t} &= 2A(\sinh u(t) - x(t)\cosh u(t)), \\ \frac{\mathrm{d}y(t)}{\mathrm{d}t} &= 2A(\sinh v(t) - y(t)\cosh v(t)), \end{aligned} \tag{24.4}$$

其中 $\sinh u = (\mathrm{e}^u - \mathrm{e}^{-u})/2, \cosh u = (\mathrm{e}^u + \mathrm{e}^{-u})/2$.

我们尝试使用 MAPLE 在区间 $\langle 0, T\rangle$ 上求解 (24.4), 初始条件为 $x(0) = X, y(0) = Y$:

```
> u := am + bm*m0*x(t) + cm*n0*y(t):
> v := an + bn*m0*x(t) + cn*n0*y(t):
> eq1 := diff(x(t),t) = 2*A*(sinh(u) - x(t)*cosh(u)):
> eq2 := diff(y(t),t) = 2*A*(sinh(v) - y(t)*cosh(v)):
> init := x(0) = X, y(0) = Y:
> sol := dsolve({eq1, eq2, init}, {x(t), y(t)});
```

$$sol :=$$

上面对于一般参数 $a_m, a_n, b_m, b_n, c_m, c_n$ 的初值问题似乎没有解析解. 我们可以使用 MAPLE 计算数值解.

24.2.1 不带调节的周期迁移

移民模型 (24.3),(24.4) 的解依赖于参数的数值. 有三种不同类型的解 (参见 [1]). 其中两种收敛于定常的平衡状态 (一种是在每个居民区我们将有 m_0 个第一群人和 n_0 个第二群人, 另一种是每一群人分别占据一个居民区). 最有趣的是我们这里将要介绍的第三种解, 它将收敛于一个极限环.

Květoň 研究了转移概率的参数 $A, a_m, a_n, b_m, b_n, c_m, c_n$(参见 [2]).Květoň 研究了生活在 Most (Northern Bohemia, 捷克共和国) 的 Czechs 人和 Gipsies 人之间的移民. 基于统计学的研究 Květoň 发现了如下的参数值:

```
> param := A = 1/2, am = 0, bm = 1.2e-4, cm = 0.5e-3,
>          m0 = 10000, an = 0, bn = -1e-4, cn = 1.2e-3,
>          n0 = 1000:
```

```
> save(u, v, param, 'system.sav');
```

我们求方程组 (24.4) 的数值解，初始条件为两群人接近相等地被分在两个居民区中.

```
> init := x(0) = 0, y(0) = 0.01:
> eq1 := subs(param, eq1):  eq2 := subs(param, eq2):
> sol := dsolve({eq1, eq2, init}, {x(t), y(t)}, numeric):
> with(plots):
> odeplot(sol, [x(t), y(t)], 0..60, numpoints = 300,
>   view=[-1..1, -1..1]);
> odeplot(sol, [t,x(t), y(t)], 0..60, numpoints = 300,
>   orientation = [70,55], colour = black, axes = normal);
```

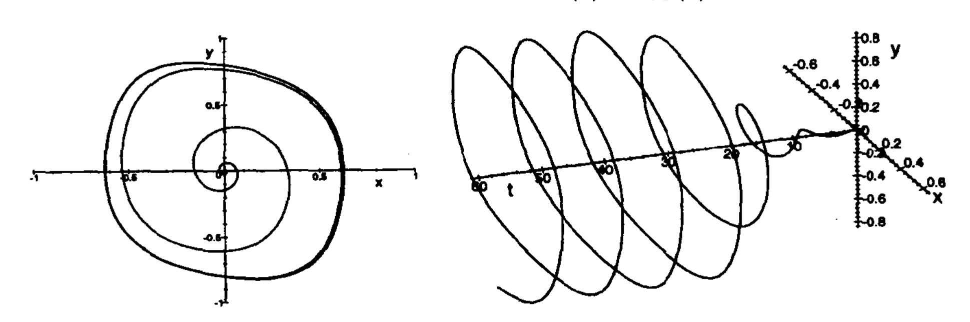
解 $x(t)$ 关于 $y(t)$ 的相平面图 (参见图 24.1 左侧) 使我们能够了解这个解如何收敛于震荡形式的周期解 (参见图 24.1 右侧).

Květoň 还发现两个人群的初始分布不会影响到他们周期地迁移. 这可以由下面的例子说明:

```
> init := x(0) = -1, y(0) = 1:
> sold := dsolve({eq1, eq2, init}, {x(t), y(t)},numeric):
> odeplot(sold, [x(t), y(t)], 0..60, numpoints = 300,
>         view = [-1..1, -1..1], color = black);
```

图 24.2 显示出 $x(t)$ 关于 $y(t)$ 的相平面图与图 24.1 左侧的相平面图有相同的函数闭包.

图 24.1 不带调节的周期迁移, $x(0) = 0, y(0) = 0.01$



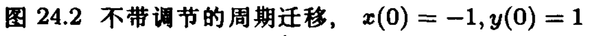
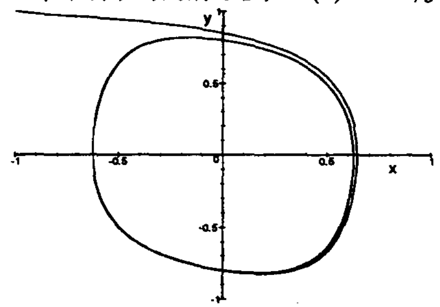


图 24.2 不带调节的周期迁移, $x(0) = -1, y(0) = 1$



这个解表明当一个人群的居民不愿意与第二个人群在同一个居民区生活时常常会发生周期性的移民.

24.2.2 带有调节的周期迁移

Weidlich 和 Haag 对于居民有向迁移发展了模型 (24.3)(24.4)(见 [6]):

$$\begin{aligned}\frac{\mathrm{d}x(t)}{\mathrm{d}t} &= 2A(\sinh u(t) - x(t)\cosh u(t)) + r,\\ \frac{\mathrm{d}y(t)}{\mathrm{d}t} &= 2A(\sinh v(t) - y(t)\cosh v(t)) + s\end{aligned}$$

其中 r 和 s 是某些调节参数 (即, 它表明政府支持某一群居民迁移到某一个指定居民区的直接干预).

调节参数 r 和 s 的作用可以使用同上节一样的数据 (见 [2]) 显示出来:

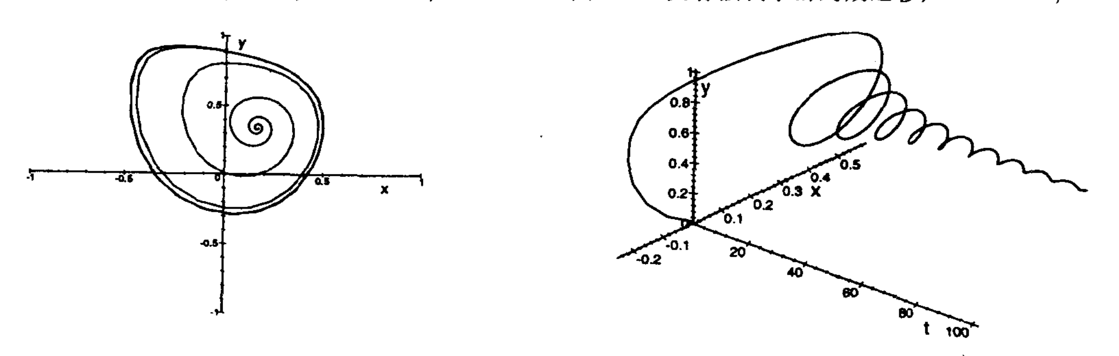
```
> restart;
> with(plots):
> read(`system.sav`):
> eq1r := diff(x(t), t) = 2*A*(sinh(u) - x(t)*cosh(u)) + r:
> eq2s := diff(y(t), t) = 2*A*(sinh(v) - y(t)*cosh(v)) + s:
> eq1r := subs(param, eq1r):  eq2s := subs(param, eq2s):
> initrs := x(0) = 0, y(0) = 0.01:
> r := -0.2:  s := 0.1:
> solrs := dsolve({eq1r, eq2s, initrs}, {x(t), y(t)}, numeric):
> odeplot(solrs, [x(t), y(t)], -30..50, numpoints = 200,
>          view = [-1..1, -1..1], color = black);
```

图 24.3 的相平面图与图 24.2 的相平面图有类似的震荡解闭包.

增大调节参数到 $r = -0.3, s = 0.3$, 问题的解收敛于一个稳定的值, 如图 24.4 所示.

```
> r := -0.3:  s := 0.3:
> solrs := dsolve({eq1r, eq2s, initrs}, {x(t), y(t)}, numeric):
> odeplot(solrs, [t, x(t), y(t)], 0..100, numpoints = 300,
>          orientation = [-50, 45], axes = normal, colour = black);
```

图 24.3 具有弱调节的周期迁移, $r = -0.2, s = 0.1$　　图 24.4 具有强调节的周期迁移, $r = -0.3, s = 0.8$



我们可以看出, 适当选择调节参数, 完全可能得到指定的移民结构.

24.3 战略投资模型

我们将考虑以 "Schumpeter 时钟" 著称的经济上的周期震荡 (萧条和繁荣周期)(参见 [4]). 在工业循环的模型中基本量是战略投资 $I(t) = E(t) + R(t)$, 其中 $E(t)$ 是扩大化投资, $R(t)$ 是合理化投资. 我们定义投资结构指数为 $Z(t) = (E(t) - R(t))/I(t)$, 我们把它分解为一个常量 Z_0 和一个变量 $z(t)$, 用它表示 Schumpeter 时钟的记时 (参见 [1]),

$$Z(t) = Z_0 + z(t).$$

组建 Schumpeter 时钟模型时要考虑到投资结构依赖于投资者的分布. 这里有两类投资者 (见 [5]). 一类是 E 型投资者, 他们偏爱累进的或扩大的投资并且投资要远大于平均投资额. 另一类是 R 型投资者, 他们偏爱合理的投资并且投资要小于平均投资额.

令 $n_1(t)$ 是 E 型投资者的数量, $n_2(t)$ 是 R 型投资者的数量, 同时所有投资者的数量是常数 $2N$, 即, $2N = n_1(t) + n_2(t)$. 使用投资者布局函数 $n(t) = (n_1(t) - n_2(t))/2$, 我们可以写出 (类似于上一节)$n_1(t) = N + n(t), n_2(t) = N - n(t)$.

在上一节引入了转移概率 $w_{ij}^{\alpha}(t)$. 在这一节, 转移概率 $w_{ij}^{\alpha}(t)$ 意味着 (每单位时间) 子系统 α 的投资者观点改变的概率, 也就是说他的隶属关系从群体 j(R 型投资者) 转变到群体 i(E 型投资者) 的概率. 对于上面定义的投资者数量 $n(t), n_1(t), n_2(t)$ 使用方程 (24.1), 我们得到下面的投资者分布的平衡方程

$$\frac{\mathrm{d}n(t)}{\mathrm{d}t} = n_2(t)w_{21}(t) - n_1(t)w_{12}(t). \tag{24.5}$$

把新的标准化变量 $x(t) = n(t)/N$ 代入方程 (24.5), 就得到如下的方程

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = w_{21}(t)(1 - x(t)) - w_{12}(t)(1 + x(t)). \tag{24.6}$$

主要的问题仍然是求出转移概率 $w_{12}(t), w_{21}(t)$ 的适当公式 (见 [3]). 用转换函数 $p(t)$ 表示投资者的主张从一个类型到第二个类型的变化. $p(t)$ 的正值表示从 R 型投资到 E 型的转变. 反之, $p(t)$ 为负值. 协调量 q 表示一个投资者与另一个投资者努力同步的强度. 则可以定义如下的转移概率函数:

$$\begin{aligned} w_{12}(t) &= a\mathrm{e}^{p(t)+qx(t)} \\ w_{21}(t) &= a\mathrm{e}^{-(p(t)+qx(t))} \end{aligned}$$

其中 A 是时间尺度因子.

把这些转移概率函数代入方程 (24.6), 类似于上一节, 我们得到:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = 2A[\sinh(p(t) + qx(t)) - x(t)\cosh(p(t) + qx(t))]. \tag{24.7}$$

投资震荡是由转换函数 $p(t)$ 的时间变化产生的. Mensch[3] 导出了如下的关于转换函数的方程

$$\frac{\mathrm{d}p(t)}{\mathrm{d}t} = -2B(p_0 \sinh(ax(t)) + p(t)\cosh(ax(t))), \tag{24.8}$$

其中 $B > 0, a > 0$ 和 $p_0 > 0$ 是特征常数.

方程 (24.7) 和 (24.8) 描述了一个战略投资的发展模型. 它被用于模拟德国从 50 年代到 70 年代的经济增长 (见 [3]), 实际情况和计算结果非常一致.

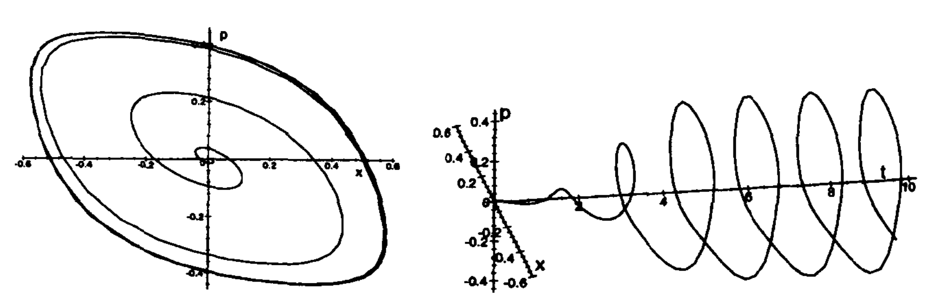
为了求解和显示 Schumpeter 时钟，我们取下述参数值： $B = 0.8, a = 4, p_0 = 0.5, A = 4B, q = 1.6$(见 [1]), 初始条件为 $x(0) = 0.01, p(0) = 0$.

```
> restart:
> eq1 := diff(x(t), t) = 2*A*(sinh(p(t) + q*x(t)) -
>        x(t)*cosh(p(t) + q*x(t))):
> eq2 := diff(p(t),t) = -2*B*(p0*sinh(a*x(t)) +
>        p(t)*cosh(a*x(t))):
> B := 0.8:  A := 4*B: a := 4:  p0 := 0.5:  q := 1.6:
> init := x(0) = 0.01, p(0) = 0:
> sol := dsolve({eq1, eq2, init}, {x(t), p(t)}, numeric):
> with(plots):
> odeplot(sol, [x(t), p(t)], 0..10, numpoints = 200,
>         color = black);
> odeplot(sol, [t, x(t), p(t)], 0..10, numpoints = 200,
>         orientation = [-100, 50], axes = normal, colour = black);
```

图 24.5 的左图显示出问题的解和极限环. 这个图可以解释为：

1. 象限 $(p(t) > 0, x(t) > 0)$ 表明当 E 型投资者为多数并且经济环境有利于扩大投资. 这时我们得到了一个繁荣的周期.
2. 象限 $(p(t) < 0, x(t) > 0)$ 表明当 E 型投资者仍然为多数，但经济环境支持到合理投资的转移. 转换函数 $p(t)$ 的值已经是负的，这意味着 E 型投资者的数量将减少. 这是一个从繁荣到萧条的转移状态.
3. 象限 $(p(t) < 0, x(t) < 0)$ 表明当 R 型投资者为多数同时经济环境有利于合理投资. 这时我们得到了经济的萧条状态.
4. 象限 $(p(t) > 0, x(t) < 0)$ 表明即使仍然是 R 型投资者为多数，趋于扩大投资的趋势已经开始了. 这个状态是从萧条转向繁荣的转移状态.

图 24.5 Schumpeter 时钟



参考文献

[1] J. KREMPASKÝ ET AL, *Synergetics*, Veda, Bratislava, 1987(in Slovak).

[2] J. KREMPASKÝ, R. KVĚTOŇ, Acta Phys. Slov., Vol.33, 1983, p.115.

[3] G. MENSCH, G. HAAG, W. WEIDLICH, Econometrica, Vol.50, 1982, p.15.

[4] J.A. SCHUMPETER, *Konjunkturzyklen, eine theoretische, historisch und statistische Analyse des kapitalistischen Prozesses. Vol.2*, Vandenhoeck and Ruprecht, Gottingen, 1961.

[5] W. WEIDLICH, G. HAAG, *Concepts and models of quantitative sociology. The dynamics of interacting populations.*, Springer Verlag, Berlin-Heidelberg-New York, 1983.

[6] W. WEIDLICH, G. HAAG, *Dynamics of interacting groups in society, in H. Haken: Dynamics of synergetics systems*, Springer Verlag, Berlin-Heidelberg-New York, 1980.

第二十五章　解析函数的等值图

W. Gautschi and J. Waldvogel

25.1　引言

在 MATLAB 中有两种构造解析函数等值图的简易方法，即等模线和等相位线. 一个是利用 MATLAB 的两变量函数 contour 命令，另一个是解等值线满足的微分方程. 这里用函数 $f(z)=e_n(z)$ 来说明，其中

$$e_n(z)=1+z+\frac{z^2}{2!}+\cdots+\frac{z^n}{n!} \tag{25.1}$$

是指数级数的前 n 项部分和. e_n 为 1 的等模线对常微分方程的数值解有特别意义，对 n 阶 Taylor 展开方法和 n 阶 $(1\le n\le 4)$ 的任意 n 步 Runge- Kutta 显式方法 (参见 [4,§9.3.2]), 它描绘了绝对稳定区域.

25.2　由 contour 命令画等值图

设 $f(z)=re^{i\varphi}$ 是解析的. 我们可以将模 r 当作两个变量 x,y 的函数，其中 $z=x+iy$; 类似地，相位 φ 也可以作为两个变量 x,y 的函数， $-\pi<\varphi\le\pi$. 于是，我们可以对 r 和 φ 应用 MATLAB 的命令 contour 而得到模和相位的等值线.

在下面的 MATLAB 程序中， x 和 y 值的集合被集中在矩阵 a 中 (在 MATLAB 意义下), 作为 contour 命令的输入矩阵，以算出 $f=e_n$ 的 r 和 φ 的值.

程序首先定义等值图的网格 h 和数值 nmax. 向量 bounds 包含对整个图有效的 x 和 y 坐标的上界和下界. 这里所选取界限被用来适应前四项指数和. 然后定义 $f(z)$ 的模和相位的等值量 vabs0 和 vang0. 最后的工作是生成向量 x 和 y, 它们包含了用于网格点矩阵 a 的离散的 x 和 y 值. 在 n 次循环中， e_n 在整个网格上的值 f 是由与计算 e_n 在单点上的值几乎一样的命令生成的，其中 t 代表级数 (25.1) 的一个单项. 唯一的差别是表达式 t=t.*a/n, 其中运算符号 .* 表示矩阵 t 和 a 的元素逐个相乘. 这个程序的最后一行 (在这里被符号 % 注释掉) 产生 figure(n) 的 EPS 文件 fign.eps, 可以被打印或插入一个文本文件.

```
>> % Contour plots of the first nmax exponential sums (Figure 1)
>> %
>> h = 1/64;  nmax = 4;  bounds = [-3.25  .75  -3.375  3.375];
>> vabs0 = [0:.1:1];  vang0 = [-.875:.125:1]*pi;
>> x = bounds(1):h:bounds(2);  y=bounds(3):h:bounds(4);
>> a = ones(size(y'))*x + i*y'*ones(size(x));
>> %  Next line: a shorter way of generating a (more memory!)
>> %  [xx,yy]=meshgrid(x,y); a=xx+i*yy;
>> t = ones(size(a));  f = t;
>> for n = 1: nmax
```

```
>>    if n <= 2, vabs = vabs0; vang = vang0;
>>      elseif n == 3, vabs = [vabs0 .47140452]; vang = vang0;
>>      else vabs = [vabs0 .58882535 .27039477];
>>           vang = [vang0 1.48185376 -1.48185376];
>>    end;
>>    t = t.*a/n;  f = f + t;
>>    figure(n);  clf;  hold on;
>>    axis(bounds);  axis image;
>>    contour(x, y, abs(f), vabs);
>>    contour(x, y, angle(f), vang);
>> end;
>> % figure(n); print -deps fign;
```

$n=1:1:4$[1] 的结果展示在下面的平面图中. e_n 的 n 个零点清晰可见，从这些零点发射出等相位线. 在这些零点附近，等模线形成环状，半径趋于零，逼近零点. 等值线对应 $r=.1:.1:1$ 和 $\varphi=\frac{7}{8}\pi:\frac{1}{8}\pi:\pi$.

在 z_0 点，$e_n'(z_0)=e_{n-1}(z_0)=0, n\geq 2$, 两条等模线相交 (参见下面 §3.1).$r$ 值分别为 $r=|e_n(z_0)|$ 或 $r=|z_0|^n/n!$, 这是因为

$$e_n(z)=e_{n-1}(z)+\frac{z^n}{n!}. \tag{25.2}$$

这些临界线也出现在平面图中 (见程序的 `if` 命令). 当 $n=2$ 时，这些曲线穿过 $z_0=-1$, 相应的 $r=\frac{1}{2}$, 而对 $n=3$ 和 $n=4$, 分别有 (精确到 8 位十进制数):$z_0=-1\pm\mathrm{i}$, $r=\sqrt{2}/3=.47140452$ 和 $z_0=-.70196418+1.80733949\mathrm{i}$, $r=(1.93887332)^4/24=.58882535$, $z_0=-1.59607164$, $r=.27039477$.

好的 r 曲线一定也对应好的 φ 曲线！它们的奇异点也是 e_n' 的零点 z_0 (参见 §3.2), 在这一点相应的 φ 值由 $e_n(z_0)/|e_n(z_0)|=(z_0/|z_0|)^n=\mathrm{e}^{\mathrm{i}\varphi}$ 定义，即 $\varphi=n\arg z_0$. 因此，对 $n=2$, 我们有 $\varphi=0\ (\mathrm{mod}\ 2\pi)$, 而对 $n=3$ 我们得到 $\varphi=\pm\frac{\pi}{4}$, 分别对应 $z_0=-1\pm\mathrm{i}$. 所有这三个 φ 值已被包含在上面列出的值中. 对 $n=4$, 在前面段落已给出的 z_0 的两个复值产生 $\varphi=4\arg z_0=\pm1.48185376(\mathrm{mod}\ 2\pi)$, 并且 z_0 的实值给出 $\varphi=0$. 这些临界的 φ 曲线也出现在图 25.1 中.

为了得到好的分辨率，采用步长 `h=1/64` 生成图像，甚至对相应于 `|angle(f)|=pi` 的“分支切割”也采用同样的步长. 选择 `h=1/32` 也可以，而 `h=1/16` 也能迅速地生成令人满意的图形.

25.3 微分方程

对任意解析函数 f, 设

$$w=f(z),\qquad w=r\mathrm{e}^{\mathrm{i}\varphi},\qquad z=x+\mathrm{i}y. \tag{25.3}$$

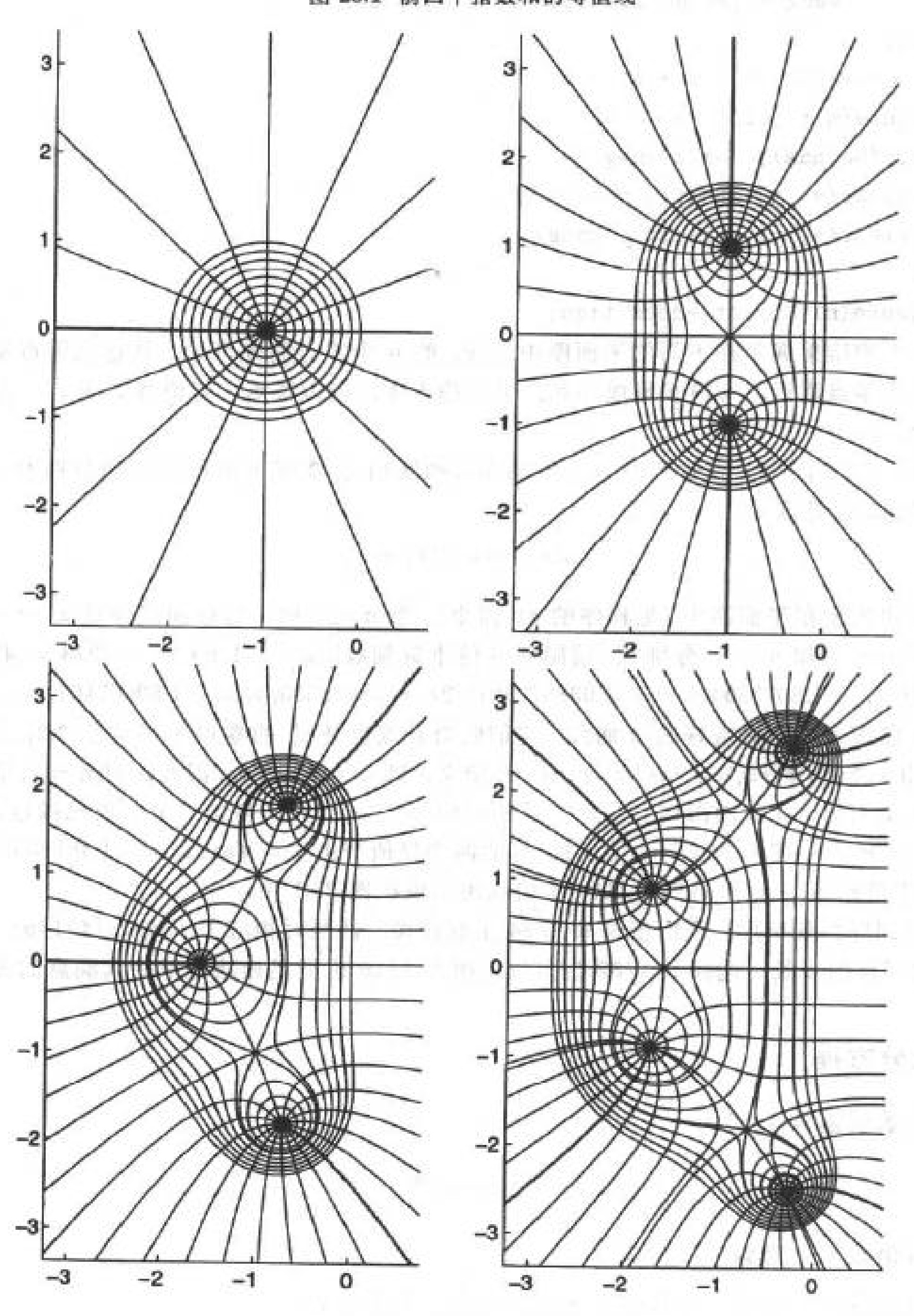
25.3.1 等值线 $r=$ 常数

为描述曲线 $r=$ 常数，自然取 φ 为独立变量. 关于 φ 对

$$f(z(\varphi))=r\mathrm{e}^{\mathrm{i}\varphi},\qquad r=\text{常数}, \tag{25.4}$$

[1]这是 MATLAB 表示 $n=1,2,3,4$ 的记号

图 25.1 前四个指数和的等值线



求微分则得到 $f'(z)\frac{dz}{d\varphi}=ire^{i\varphi}=if(z)$, 即

$$\frac{dz}{d\varphi}=iq(z),\qquad 其中\qquad q(z)=\frac{f(z)}{f'(z)}. \tag{25.5}$$

用 s 表示弧长，则有

$$\frac{ds}{d\varphi}=\sqrt{(\frac{dx}{d\varphi})^2+(\frac{dy}{d\varphi})^2}=\left|\frac{dz}{d\varphi}\right|=|q(z)|,$$

于是

$$\frac{dz}{ds}=\frac{dz}{d\varphi}\frac{d\varphi}{ds}=i\frac{q(z)}{|q(z)|}. \tag{25.6}$$

写成微分方程组，就是

$$\begin{aligned}\tfrac{dx}{ds}&=-\mathrm{Im}\left\{\tfrac{q(z)}{|q(z)|}\right\},\\ \tfrac{dy}{ds}&=\mathrm{Re}\left\{\tfrac{q(z)}{|q(z)|}\right\},\end{aligned}\qquad z=x+iy. \tag{25.7}$$

如果我们对穿过实轴的等值线感兴趣，我们必须为 (25.7) 找一个初始点 $x(0)=x_r$, $y(0)=0$, 取实数 x_r 使得 $f(x_r)=r$(假设对实数 x, $f(x)$ 是实值). 当 $f(x)=e_n(x)$ 时，如果 $r\geq 1$, 这一点容易做到，因为 $e_n(0)=1$ 且 $e_n(x)$ 对 $x\geq 0$ 单调增. 所以存在唯一的 $x_r\geq 0$ 使得 $e_n(x_r)=r$. 如果 $0<r<1$, 当 n 是奇数时，这一点仍可以做到. 那么 $e_n'(x)=e_{n-1}(x)>0$, 因为当 m 是偶数时，已知 e_m 的所有零点都是复数 [3](也可以参见 [1]). 于是当 x 从 $-\infty$ 增到 ∞ 时， e_n 从 $-\infty$ 单调地增到 ∞, 且存在唯一的 $x_r<0$ 使得 $e_n(x_r)=r$. 当 n 是偶数，对一切实的 x 都有 $e_n(x)>0$, 而且 $e_n'=e_{n-1}$ 仅在一点 $x_0<0$ 为零， e_n 在这一点取最小值 (参见 [2]). 由 (25.2) 和 $e_{n-1}(x_0)=0$, 我们得到 $e_n(x_0)=x_0^n/n!$, 并且得知当且仅当 $r\geq x_0^n/n!$ 时 $e_n(x_r)=r$ 有一个解 $x_r<0$. 对更小的正数 r, 则必须在 e_n 的某个复零点附近找到一个复的初始点 $x(0)$, $y(0)>0$.

使 $f'(z_0)=0$ 的 z_0 点是 (25.7) 的一个奇异点，在这一点两条 r 线相交成直角. 这要求特别小心地对 (25.7) 在四个方向上求积分. 初始点当然是 z_0, 即 $x(0)=\mathrm{Re}\,z_0$, $y(0)=\mathrm{Im}\,z_0$. 需要分析 (25.7) 的右边在 z_0 的值. 设 $h(z)=(z-z_0)q(z)$; 则 h 在 z_0 附近是光滑的且有 Taylor 展式

$$h(z)=\frac{f_0+\frac{1}{2}(z-z_0)^2f_0''+\cdots}{f_0''+\frac{1}{2}(z-z_0)f_0'''+\cdots},\qquad h(z_0)=\frac{f_0}{f_0''},$$

其中 $f_0=f(z_0)$(我们假设 $f_0\neq 0$ 且 $f_0''\neq 0$). 设

$$\frac{z-z_0}{|z-z_0|}=e^{i\theta},\ -\frac{1}{2}\pi<\theta\leq\frac{1}{2}\pi,\ h(z_0)/|h(z_0)|=e^{i\omega},\ -\pi<\omega\leq\pi,$$

(注意到对每个 θ 存在 $\theta+\pi$ 对应于曲线向后的延续), 则我们有

$$\frac{q(z)}{|q(z)|}=\frac{|z-z_0|}{z-z_0}\frac{h(z)}{|h(z)|}\to e^{i(\omega-\theta_0)}\qquad 当 z\to z_0,$$

其中 $\theta_0=\lim_{z\to z_0}\theta$. 还需确定 θ_0.

沿着过点 z_0 的 r 曲线，我们有

$$\begin{aligned}r^2&=|f(z)|^2=|f_0+\tfrac{1}{2}(z-z_0)^2f_0''+\cdots|^2\\&=|f_0|^2+\mathrm{Re}[(z-z_0)^2f_0''\bar{f}_0]+O(|z-z_0|^3).\end{aligned}$$

因为 $|f_0|^2 = r^2$, 所以

$$\mathrm{Re}\left\{\left(\frac{z-z_0}{|z-z_0|}\right)^2 f_0''\bar{f}_0\right\} = O(|z-z_0|),$$

当 $z \to z_0$ 时,

$$\mathrm{Re}(\mathrm{e}^{2\mathrm{i}\theta_0} f_0''\bar{f}_0) = 0.$$

因此

$$\tan 2\theta_0 = \frac{\mathrm{Re}(f_0''\bar{f}_0)}{\mathrm{Im}(f_0''\bar{f}_0)}. \tag{25.8}$$

在 $-\frac{1}{2}\pi < \theta_0 \le \frac{1}{2}\pi$ 中，确实存在两个解，它们之差为 $\frac{1}{2}\pi$, 从而证实了过 z_0 点的两条 r 曲线的直交性.

注意到当 $f(z) = e_n(z)$ 时，有 $f'(z) = e_{n-1}(z)$, 因此 z_0 是 e_{n-1} 的一个零点. 这在 §2 的平面图上可以清楚地看到. 而且，如果 $n \ge 2$, 那么 $f_0 = e_n(z_0)$, $f_0'' = e_{n-2}(z_0)$, 于是在 (25.2) 中令 $z = z_0$, 再用 $n-1$ 代替 n, 得到 $f_0 = z_0^n/n!$, $f_0'' = -z_0^{n-1}/(n-1)!$, 推导出关于 θ_0 的方程

$$\tan 2\theta_0 = -\frac{\mathrm{Re}\, z_0}{\mathrm{Im}\, z_0} \qquad (f = e_n).$$

25.3.2 等值线 $\varphi =$ 常数

对曲线 $\varphi =$ 常数，我们取 r 为自变量，对

$$f(z(r)) = r\mathrm{e}^{\mathrm{i}\varphi}, \qquad \varphi = \text{常数},$$

关于 r 微分，得到

$$\frac{\mathrm{d}z}{\mathrm{d}r} = \frac{\mathrm{e}^{\mathrm{i}\varphi}}{f'(z)}.$$

用 s 表示弧长，则

$$\frac{\mathrm{d}s}{\mathrm{d}r} = \left|\frac{\mathrm{d}z}{\mathrm{d}r}\right| = \frac{1}{|f'(z)|},$$

因此

$$\frac{\mathrm{d}z}{\mathrm{d}s} = \frac{\mathrm{d}z}{\mathrm{d}r}\frac{\mathrm{d}r}{\mathrm{d}s} = \mathrm{e}^{\mathrm{i}\varphi}\frac{|f'(z)|}{f'(z)},$$

或者，写成微分方程组

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}s} &= \mathrm{Re}\left\{\mathrm{e}^{\mathrm{i}\varphi}\frac{|f'(z)|}{f'(z)}\right\}, \\ \frac{\mathrm{d}y}{\mathrm{d}s} &= \mathrm{Im}\left\{\mathrm{e}^{\mathrm{i}\varphi}\frac{|f'(z)|}{f'(z)}\right\}, \end{aligned} \qquad z = x + \mathrm{i}y. \tag{25.9}$$

(25.9) 的奇异点 z_0 仍是 f' 的零点. 在这一点

$$\frac{f_0}{|f_0|} = \mathrm{e}^{\mathrm{i}\varphi}, \qquad -\pi < \varphi \le \pi,$$

确定了 φ. 当 $z \to z_0$ 时，$|f'(z)|/f'(z)$ 的极限可以由类似 §3.1 中的过程确定. 但在这里，我们直接用 f 在 z_0 点的 Taylor 展式，以便研究在 $f'(z_0) = 0$ 的点 z_0 附近的 φ 曲线 (对 r 曲线也一样). 设 $z = z_0 + \zeta$, ζ 为复增量，并记 f 在 z_0 的导数为

$$f_k := f^{(k)}(z_0), \qquad k \ge 0, \qquad f_0 \ne 0,\ f_1 = 0,\ f_2 \ne 0. \tag{25.10}$$

则 Taylor 级数为

$$f(z_0+\zeta)=f_0+f_2\frac{\zeta^2}{2!}+f_3\frac{\zeta^3}{3!}+\cdots. \tag{25.11}$$

下一步，在 (25.3) 中定义 $w=f_0\mathrm{e}^u$, 即令

$$f(z_0+\zeta)=f(z_0)\mathrm{e}^u, \tag{25.12}$$

我们看到，过 z_0 点的 r 曲线由对应纯虚数 $u=it$ 的 ζ 值给出，而过 z_0 点的 φ 曲线由 $u\in\mathbf{R}$ 给出． z_0 点本身对应于 $\zeta=u=0$. 因此我们需从方程 (25.12) 中解出 ζ, 其中的 $f(z_0+\zeta)$ 用 (25.11) 式代替，解这个方程是 MAPLE 的一个特长．

在下面程序 [2] 中，级数 (25.11) 和方程 (25.12) 将分别记为 s 和 eq. 命令 solve 自动地将 e^u 展成 Taylor 级数并借助于一系列的递推，按 u 的适当幂 (这里用半整数幂) 解方程．正如所希望的那样，对应两个平方根可能值的两个解被解得．只对第一个解 zet0[1] 进一步处理：首先定义在方程 (25.10) 中的缩写 fk 被取代，然后根据

$$u=\frac{v^2f_2}{2f_0}\quad\text{或}\quad v=\left(\frac{2uf_0}{f_2}\right)^{\frac{1}{2}} \tag{25.13}$$

引入变量 v. 符号 D(f) 和 (D@@k)(f) 分别表示 f 的导数和 f 的 k 阶导数．用函数 map 引入运算，即将函数的第一个变量，这里是根的简称，作用到由第二个变量给出的表达式中的每一项．最后，用 series 对 O 项简化．

```
> N := 5:  Order := N:
> s := series(f(z0 + dz), dz):
> s0 := subs(D(f)(z0) = 0, s);
```

$$s0:=\mathrm{f}(z0)+\frac{1}{2}(D^{(2)})(f)(z0)\,dz^2+\frac{1}{6}(D^{(3)})(f)(z0)\,dz^3+\frac{1}{24}(D^{(4)})(f)(z0)\,dz^4+\mathrm{O}(dz^5)$$

```
> eq := s0 = f(z0)*exp(u):
> zet0 := solve(eq, dz);
```

$$zet0:=\frac{\mathrm{f}(z0)\sqrt{2}\sqrt{u}}{\sqrt{\%1}}-\frac{1}{3}\frac{(D^{(3)})(f)(z0)\,\mathrm{f}(z0)\,u}{(D^{(2)})(f)(z0)^2}+\frac{1}{288}$$
$$(40\,(D^{(3)})(f)(z0)^2\,\mathrm{f}(z0)^3-24\,(D^{(2)})(f)(z0)\,\mathrm{f}(z0)^3\,(D^{(4)})(f)(z0)+72\,(D^{(2)})(f)(z0)^3\,\mathrm{f}(z0)^2)\sqrt{2}$$
$$u^{3/2}\Big/((D^{(2)})(f)(z0)^2\,\%1^{3/2})+\mathrm{O}(u^2),-\frac{\mathrm{f}(z0)\sqrt{2}\sqrt{u}}{\sqrt{\%1}}-\frac{1}{3}\frac{(D^{(3)})(f)(z0)\,\mathrm{f}(z0)\,u}{(D^{(2)})(f)(z0)^2}-\frac{1}{288}$$
$$(40\,(D^{(3)})(f)(z0)^2\,\mathrm{f}(z0)^3-24\,(D^{(2)})(f)(z0)\,\mathrm{f}(z0)^3\,(D^{(4)})(f)(z0)+72\,(D^{(2)})(f)(z0)^3\,\mathrm{f}(z0)^2)\sqrt{2}$$
$$u^{3/2}\Big/((D^{(2)})(f)(z0)^2\,\%1^{3/2})+\mathrm{O}(u^2)$$
$$\%1:=\mathrm{f}(z0)\,(D^{(2)})(f)(z0)$$

```
> zet1 := subs(seq( (D@@k)(f)(z0) = f.k, k=0..N-1), zet0[1]):
> zet2 := map(radsimp, subs(u = v^2*f2/2/f0, zet1)):
> zeta := series(zet2, v);
```

$$\zeta:=v-\frac{1}{6}\frac{f3}{f2}v^2+\frac{1}{72}\frac{5\,f3^2\,f0-3\,f2\,f0\,f4+9\,f2^3}{f0\,f2^2}v^3+\mathrm{O}(v^4)$$

---

[2]作者感谢 Dominik Gruntz 提供了这个程序.

MAPLE 程序对任意 $N \geq 3$ 有效，产生以上级数的 $N-2$ 项. 但是它运行得相当慢，因为它没有“智能”，如在所得级数的形式上建立信息. 但是，能够找到这个级数对一般用途的符号处理器已经相当不错了. 我们看到 ζ 可以被写作由 (25.1 3) 定义的变量 v 的形式幂级数. 如果原级数 (25.11) 在 z_0 点的某个领域内收敛，则所得级数在 $v=0$ 的某个领域内收敛.

r 曲线在 z_0 点的 θ_0 方向由对应于 $u=it$ 当 $t\to 0$ 时的 ζ 给出. 由上面的级数和方程 (25.13) 马上可以推导出

$$\theta_0 = \arg v = \frac{1}{2}(\arg f_0 - \arg f_2 \pm \frac{\pi}{2}).$$

因此过 z_0 的两条 r 曲线垂直相交，与方程 (25.8) 完全一致.

从另一方面，过 z_0 点的 φ 曲线的方向由 (25.13) 取 μ 的实数值给出. 我们得到两个方向 $\theta_0 \pm \frac{\pi}{4}$，即过 z_0 点的两条 φ 曲线的切线是 r 曲线切线的角平分线.

25.4 $f = e_n$ 的等值线 $r = 1$

正如在 §3.1 节指出的那样，我们要解 (25.7)，取初值为 $x=y=0$. 设 S_n 是 e_n 的 1 曲线与负实轴的交点. 由对称性，仅需计算每条 1 曲线在上半复平面的部分.

我们将讨论这个过程的两种方法：首先是仅使用对独立变量计算前的 s_f 值 (象在原来的 MATLAB 4 版本中一样) 进行数值积分的结果. 在这一节的末尾，我们将提出一个简化的算法，利用了现行的 MATLAB 5 版本的 Events 能力. 对两种方法都需要确定在 1 曲线上原点与 S_n 之间弧长的上界 s_f.

可以按下面的方法确定这样的上界 s_f. 首先考察当 $n\to\infty$ 时区域 $|e_n(z)|\leq 1$ 与半径为 $\rho(n)$ 的半圆的逼近. 由渐近分析得

$$\rho(n) = \exp(-1)\cdot(n + \log\sqrt{2\pi n} + O(1)).$$

凭经验选择 $O(1)$ 为 3; 于是得到

$$s_f = (1+\frac{\pi}{2})\exp(-1)\cdot(n+\log\sqrt{2\pi n}+3) \tag{25.14}$$

为当 $n\geq 1$ 时到 S_n 点的弧长的上界.

如果不用 Events 功能，可先对微分方程积分，直到求得最终值 s_f. 然后，仅需描出满足 $y\geq 0$ 的点. 对 1 曲线上最接近 S_n 的两点进行线性插值可以逼近点 S_n.

下面的 MATLAB 程序用带有参数 $y\geq 0$ 的 `find` 命令找到所有满足条件 $y\geq 0$ 的点集. 它们的下标被收集在向量 `indices` 里. 用在线性插值中的点的下标便是 `l=length(indices)` 和 `l1=l+1`. 最后，`w` 是含有两个插值权重的标准化的行向量，实际的插值由乘积 `w*z(l:l1,:)` 执行.

```
% Level curves r = 1 for the first 21 exponential sums (Fig. 2)
>> global n
>> nmin = 1;  nmax = 21;  tol = 3.e-8;
>> axis equal;  hold on; % arc = [];
>> options= odeset('RelTol',tol,'AbsTol',tol);
>> for n = nmin:nmax,
>>    sf = 0.94574*(n +.5*log(2*pi*n) + 3);
```

```
% 0.94574=(1+pi/2)*exp(-1)
>>    [s,z] = ode45('level4', [0 sf], [0; 0], options);
>>    indices = find(z(:, 2) >= 0);
>>    l = length(indices); l1 = l + 1;
>>    w = [-z(l1, 2), z(l, 2)]; w = w/sum(w);
>>    z(l1,:) = w*z(l:l1,:);
>>    plot(z(1:l1, 1), z(1:l1, 2));
% approximate arclengths and bounds sf
>>    % arc = [arc; w*s(l:l1), sf];
>> end;
```

算法 25.1 level4.m

```
function zdot = level4(s, z)
%LEVEL4 generates the right-hand side of
%       the differential equation for the 1-lines of
%       f(z) = 1 + z + z^2/2! + ... + z^n/n!

global n

zc = z(1) + i*z(2);  t = 1;  f0 = 0; f = t;
for k = 1: n, t = t*zc/k; f0 = f;  f = f + t; end;
q = f/f0; zdot = [-imag(q); real(q)]/abs(q);
```

MATLAB 程序先定义参量 nmin, nmax 和误差范围 tol, 借助 odeset, 它们被用来定义积分器的选项结构, options. 在 nmin $\leq n \leq$ nmax 的范围内 1 等值线在由 tol 给定的精确度内绘出. 对新的 nmin 和 nmax 的值再运行这个程序, 图形上出现新的曲线. 由注释符号 % 关闭的语句生成 arc 表, 它包含实际弧长和由 (25.14) 算出的上界 s_f.

应用积分器 ode45 执行积分. 这个过程输入的参数是: 根据方程 (25.7) 定义在算法 25.1 中的 M 文件的文件名 'level4'(在引号中), 包含了自变量的初值 0 和终值 sf 的一个向量, 初值的列向量 [0;0], 和选项结构, options. 选择 tol=3.0e-8 生成一个高分辨率的图, 而缺省 RelTol=1.0e-3, AbsTol=1.0e-6(当参量 options 在调用中被省略时) 仍可产生一个令人满意的图. 由积分器产生的自变量和因变量的值被分别储存在向量 s 和 z 中, 以备绘图使用.

$n = 1:1:21$ 的结果显示在下面的图 25.2 中. 在 n 稍稍超过 5 时, 虚轴附近变换到圆部分的图形似乎显示出周期性. 例如, 对应 $n = 5, 10, 15, 21$ 的曲线都表现出到右半平面的较大的突出.

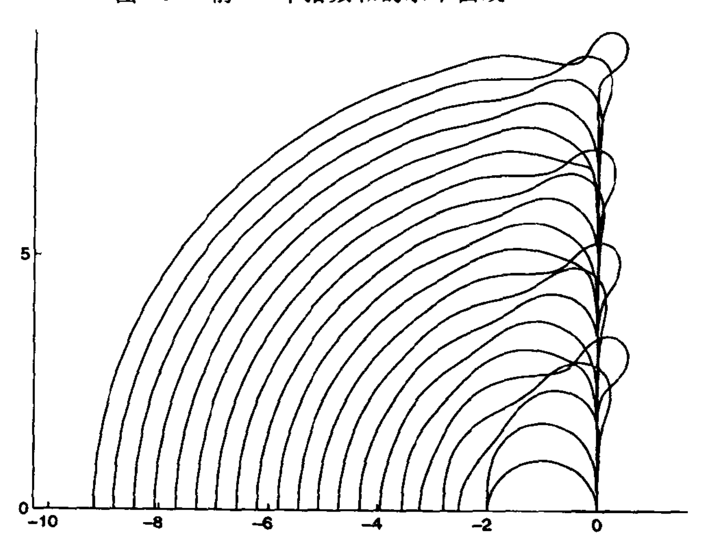
研究这个现象是有趣的, 但是超出了这篇文章的范围. 我们仅指出当 $n \to \infty$ 时, 这个周期趋于

$$\frac{2\pi}{\frac{\pi}{2} - \exp(-1)} = 5.22329130.$$

这个结果可以这样得到: 对实数 ν 考虑函数 $e_\nu(z)$(它为不完全的 Gamma 函数) 并要求 1 曲线包含一个使 $e'_\nu(z) = 0$ 的鞍点.

MATLAB 的 Events 能够在 "事件" 发生时终止数值积分, 即, 一个所谓 "事件函数" 穿过零

图 25.2 前 21 个指数和的水平曲线 $r = 1$



点. 在这里复因变量的虚部 imag(z) 即为事件函数, 它必须定义在函数 level 中 (算法 25.2), 函数 level 也定义了微分方程的右边 zdot. 这个函数需要附加的输入参数 flag 和输出参数 isterminal, 以及 direction, 使得如果 flag 被遗漏或未定义, zdot 是微分方程右边的向量. 但是, 如果 flag 的值为 'events', zdot 向量必须被定义为事件函数, 且参量 isterminal(zdot 的有关分量的下标) 和 direction(零通道的方向) 必须被适当地定义. 因为 MATLAB 5 的 ode45 能够求复值因变量的积分, 下面的程序被大大简化了.

```
% Level curves r = 1 for the first 21 exponential sums (Fig. 2)
% using the 'Events' capability and integration of complex
% dependent variables
>> global n
>> nmin = 1;  nmax = 21;  tol = 1e-6;
>> axis equal;  hold on; % arc = [];
>> options= odeset('RelTol',tol,'AbsTol',tol,'Events','on');
>> for n = nmin:nmax,
>>     sf = 0.94574*(n +.5*log(2*pi*n) + 3);
>>     [s,z] = ode45('level', [0 sf],0,options);
>>     plot(z);
>> % arclengths and bounds sf
>>     % arc = [arc; s(length(s)), sf];
>> end;
```

算法 25.2 level.m

```
function [zdot, isterminal, direction] = level(s, z, flag)
%LEVEL generates the right-hand side of
```

```
%      the differential equation for the 1-lines of
%      f(z) = 1 + z + z^2/2! + ... + z^n/n!

global n

if nargin<3 | isempty(flag),
   t = 1;  f0 = 0; f = t;
   for k = 1:n, t = t*z/k; f0 = f;  f = f + t; end;
   q = f/f0; zdot = i*q/abs(q);
else
   switch(flag)
   case 'events'
      zdot= imag(z);
      isterminal= 1;
      direction= -1;
   otherwise
      error(['Unknown flag: ', flag]);
   end;
end
```

25.5 $f = e_n$ 的等值线 $\varphi =$ 常数

下面是一个 MATLAB 程序，对任意固定的 $n > 0$ 执行 §3.2 的方法，这里微分方程 (25.9) 必须在函数 phase 中实现，并存储在 M 文件 phase.m (算法 25.3) 中.

```
% Lines of constant phase for the 10th exponential sum (Fig 3)
%
>> global n phi
>> n = 10; tol = 1.e-5; sf = 1.5;
>> clf; axis([-6 6 -1 8]); hold on;
>> r = roots(1./gamma(n + 1:-1:1));
>> indices = find(imag(r) >= 0); zero = r(indices)
>> options= odeset('RelTol',tol,'AbsTol',tol);
>> for k = 1: length(zero),
>>    z0 = zero(k);
>>    for phi = -7/8*pi:pi/8:pi,
>>       [s,z] = ode45('phase', [0 sf], z0, options);
>>       plot(z);
>>    end;
>> end;
```

程序先定义 n, 误差范围 tol, 和期望的从零点发出的曲线段的弧长 sf. 然后借助函数 roots 计

算 e_n 的零点的向量 r, 其中 e_n 的系数由 Gamma 函数生成. 在下一行, 借助以 `imag(r)>=0` 为参数的 `find` 命令形成上半平面内零点的子集. 程序中所用的指令将所有定义子集的下标存于向量 `indices`; 于是 `r(indices)` 是在上半平面 e_n 的零点的向量 (为方便起见将其打印出).

算法 25.3 phase.m

```
function zdot = phase(s,z)

global n phi

eiphi = exp(i*phi);
t = 1; f = t;
for k=1:n-1, t = t*z/k; f = f+t; end;
zdot = eiphi*abs(f)/f;
```

在调用积分器 ode45 中, 输入参数是: 定义在算法 25.3 中对应方程 (25.9) 的名为 'phase'(在引号中) 的微分方程, 一个包含自变量的初值 0 和终值 sf 的向量, 复初值 z0, 和选项结构 options(省缺时为 10^{-3}, 10^{-6}). 由积分器产生的自变量和因变量的值分别存于向量 s 和 z 中, 以备作图使用. $n = 10$ 的结果描绘在图 25.3[3] 中.

图 25.3 第 10 个指数和的等相位线

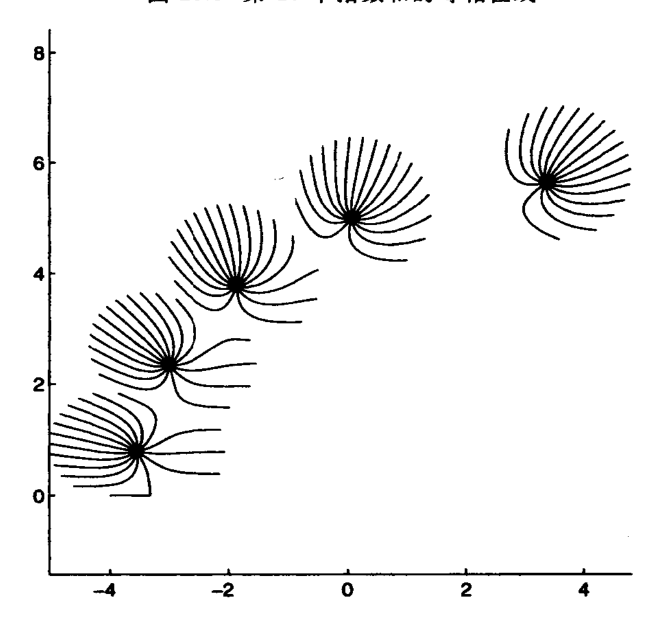


图 25.3 并未出现复奇异点, 因为所选的 φ 值不对应过复奇异点的等值线. 但是 $z_0 = -3.333551485$ 明显是一个实奇异点, 因为曲线 $\varphi = 0$(靠近图形的下部) 突然出现一个右拐的角. 令人好奇的是, 积分器 ode45 如何进行具有奇异性的积分, 它似乎做到了这一点.

[3]我们写出并运行这个程序是在 1996 年 8 月 1 日, 当时正在为庆祝瑞士国庆节放焰火.

参考文献

[1] A. J. CARPENTER, R. S. VARGA, J. WALDVOGEL, *Asymptotics for the zeros of the partial sums of* e^z. *I*, Rocky Mountain J. Math., 21,1991, pp.99-120.

[2] K. E. IVERSON, *The zeros of the partial sums of* e^z, Math. Tables and Other Aids to Computation, 7,1953, pp.165-168.

[3] G. PÓLYA, G. SZEGÖ, *Problems and Theorems in Analysis*, Vol. II, Part V, Exercise 74, Springer-Verlag, New York, 1976.

[4] H. R. SCHWARZ, *Numerical Analysis. A Comprehensive Introduction*, John Wiley & Sons, Chichester, 1989.

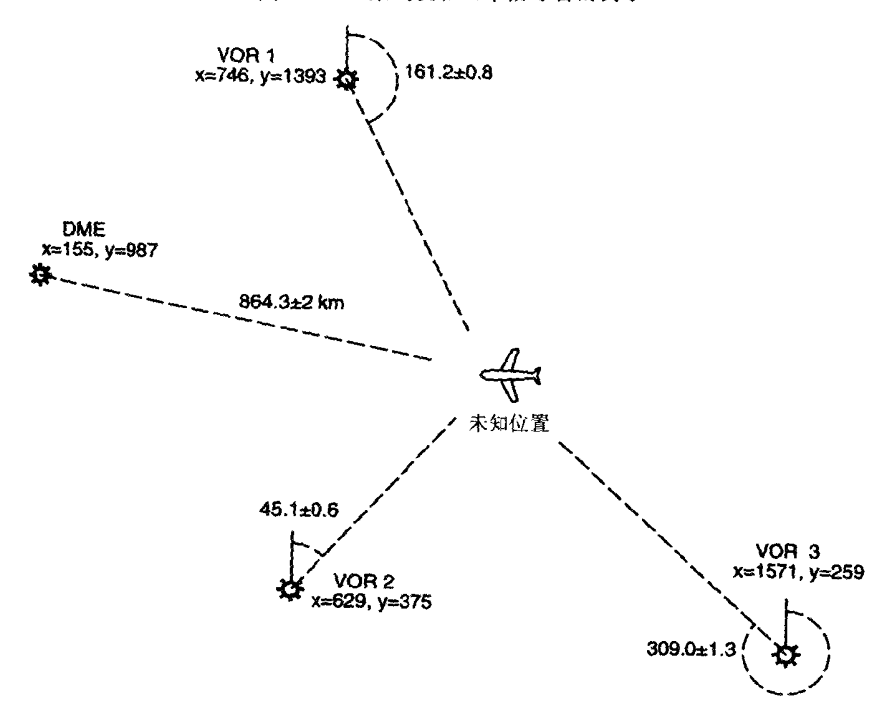
第二十六章　非线性最小二乘法：飞机的最准确的定位

G. Gonnet

26.1　引言

图 26.1 给出了一个简化的近代飞机飞行的典型状况．一架飞机处于一个未知的位置并且收到了从不同信号台发来的信号．这一章的主要目的是组建一个模型以获得的信息计算飞机的最准确的位置．

图 26.1　一架飞机和四个信号台的例子



三个信号台是 VOR 型的 (Very high frequecy OmniRange).VOR 型信号台允许飞机得到从信号台到飞机的方位角．换句话说 $\theta_1, \theta_2, \theta_3$ 对飞机是已知的． DME 型信号台 (Distance Measuring Equipment) 通过由它发出并被反射回来的信号，测量从飞机到信号台的距离．在本例中，距离是 864.3 ± 2km．

每一个测量值及其误差被给出了．测量值的标准表示是 $m \pm n$. 这意味着被测量的真值在 $m+n$ 和 $m-n$ 之间．不同的学科对于术语"在 $\cdots$ 之间"有不同的解释．它可能是一个绝对的陈述，即，真值总是在两个界限之间，或者是一个统计学的陈述，即，真值以 $z\%$ 的概率在两个界限之间．通常我们假设误差服从正态分布，其均值是 m，标准差为 n. 对于我们的分析，使用哪一个误差的定

义是无所谓的，但规定所有的测量值都使用同一个定义.

我们将简化所讨论的问题，只考虑二维的情况. 我们将不考虑高度，高度的数据可以由另外的仪器得到，并且它将使我们的例子更加复杂，这是不必要的.

我们用 x 和 y 表示飞机的未知坐标. 容易看出，任何一对 VOR/DME 的读数都将给出我们计算 x 和 y 的足够信息. 对于四组数据来说，这个问题是超定的. 因为测量值不是精确的，我们希望使用所有有用的信息来计算 x 和 y, 以得到更准确的结果.

输入数据列在如下的表中.

	x 坐标	y 坐标	测量数值	误差
VOR 1	$x_1 = 746$	$y_1 = 1393$	$\theta_1 = 161.2$	$\sigma_1 = 0.8$
VOR 2	$x_2 = 629$	$y_2 = 375$	$\theta_2 = 45.10$	$\sigma_2 = 0.6$
VOR 3	$x_3 = 1571$	$y_3 = 259$	$\theta_3 = 309.0$	$\sigma_3 = 1.3$
DME	$x_4 = 155$	$y_4 = 987$	$d_4 = 864.3$	$\sigma_4 = 2.0$
飞机	x	y		

这一章的结构如下. 第一部分将分析如何提出问题. 第二节将进行必要的计算来求得最优的解. 最后一节将分析结果的信度.

26.2 组建最小二乘方程

在误差服从正态分布的假设下，使用极小化误差平方和的方法来求解 x 和 y 的定位问题是完全合适的. 另一方面，如果我们不知道任何关于误差单个分布的信息，极小化误差平方和仍然是一个不错的稳定方法. 所以无需进一步的讨论，我们将把这个问题定为一个最小二乘问题.

对于 VOR 信号台 (1,2 或 3), 所满足的方程是

$$\tan\tilde{\theta}_i \approx \frac{x - x_i}{y - y_i}$$

其中 $\tilde{\theta}_i = 2\pi\theta_i/360$, 同时对于 DME(位于 x_4, y_4), 方程是

$$\sqrt{(x - x_4)^2 + (y - y_4)^2} \approx d_4.$$

(在航空领域，角度的标准测量是从正北开始沿顺时针方向，以度为单位. 这与三角学的测量是不同的，在三角学上，是从东方开始沿反时针方向，以弧度为单位. 因此必须注意从度到弧度的转换和选择正确的象限.)

误差是每个方程左侧和右侧之差. 虽然这对精确确定的系统是正确的，对于超定系统来说，它是不适当的.

我们容易看出，在我们如何表示这个误差上，这里有含混不清之处. 例如

$$\tan\tilde{\theta}_i - \frac{x - x_i}{y - y_i} \tag{26.1}$$

$$\tan^{-1}\frac{x - x_i}{y - y_i} - \tilde{\theta}_i \tag{26.2}$$

$$(y - y_i)\tan\tilde{\theta}_i - (x - x_i)$$

都是对于 VOR 信号站的误差的可能表达式. 对于 DME 方程也出现同样的含混之处:

$$\sqrt{(x-x_4)^2+(y-y_4)^2}-d_4$$

$$(x-x_4)^2+(y-y_4)^2-d_4^2.$$

我们该使用哪一个呢? 在回答这个问题之前, 我们要注意一个重要的技术问题. 虽然方程 (26.1) 总是正确的, 方程 (26.2) 可能不正确. 问题发生在 $\tan^{-1}$ 将返回一个主值, 这个值在 $-\pi/2$ 和 $\pi/2$ 之间. 这就带来两个问题, 第一个问题容易解决, 第二个则需要更加巧妙的处理方法. 第一个问题是转换航空角, 它在正规化后从 0 到 2π 的范围内转变为 $-\pi$ 到 π. 这可以通过将超过 π 的角减去一个 2π 来实现. 第二个问题是 $\tan^{-1}$ 将返回 $-\pi/2$ 和 $\pi/2$ 之间的数值. 这意味着相反的方向, 例如 135° 和 315° 是不能区别的. 这可能产生不能被满足的方程或者如果角度被化简到 $\tan^{-1}$ 的范围, 则多重的, 错误的解都可能产生. 为纠正这个问题我们需要分析 $x-x_i$ 和 $y-y_i$ 的符号, 有时称之为象限分析. 这是一个非常知名和共同的问题, MAPLE 中的两个变量的 arctan 函数 (在其它的语言中为 atan2) 可以做象限分析, 同时返回一个解决相反方向问题的 $-\pi$ 和 π 之间的数值. 方程 (26.2) 可以写成

$$\arctan(x-x_i,y-y_i)-\tilde{\theta}_i.$$

为计算误差我们需要选择更适当的公式. 除非还知道关于测量仪器的其它知识, 所使用的最好和最安全的方程是把误差表示为与测量值相同单位的方程. 例如, VOR 测量了角度, 因此对应的误差应该也是角度

$$\epsilon_i=\arctan(x-x_i,y-y_i)-\tilde{\theta}_i.$$

DME 测量了距离, 因此这个测量值的误差将是公里 (或是其它距离单位)

$$\epsilon_4=\sqrt{(x-x_4)^2+(y-y_4)^2}-d_4.$$

在计算误差平方和之前, 我们需要统一所有的测量值, 使它们具有相同的尺度. 这很容易由 ϵ_i 除以标准差 (即, 表示测量误差的那个数值) 得到.

一般来说, 对于一个测量值 $z=m\pm n$, 其中 z 满足一个方程 $F(z,a,b,\ldots)=0$, 令 $z=f(a,b,\ldots)$ 是 F 关于它的第一个变量的反函数. 则

$$\delta=\frac{m-f(a,b,\ldots)}{n}$$

是误差的一个标准化的度量. 对于服从正态分布的无偏误差, δ 服从 $N(0,1)$, 即平均值为 0 且方差为 1 的正态分布. 极小化变量的平方和, 即 $\sum\delta_i^2$, 与每一个误差服从相同的分布.

在我们的例子中所要极小化的平方和是

$$S=\sum_{i=1}^{3}\left(\frac{\arctan(x-x_i,y-y_i)-\tilde{\theta}_i}{\sigma_i}\right)^2+\left(\frac{\sqrt{(x-x_4)^2+(y-y_4)^2}-d_4}{\sigma_4}\right)^2.$$

相当明显, 这个问题关于未知变量 x 和 y 是非线性的. 也没有其它的误差表达式能够得出线性问题.

26.3 求解非线性系统

我们将使用 MAPLE 来求解这个问题，后面除了数值计算之外还需要进行一些符号演算. 首先我们定义输入数据. 我们用向量 X 和 Y 存储信号站的坐标，用 x 和 y 表示飞机的未知坐标.

```
> theta := array( [ 161.2, 45.10, 309.0 ] );
> sigma := array( [ 0.8, 0.6, 1.3, 2.0 ] );
> X := array( [ 746, 629, 1571, 155 ] );
> Y := array( [ 1393, 375, 259, 987 ] );
> d4 := 864.3;
```

$$\theta := [161.2, 45.10, 309.0]$$
$$\sigma := [.8, .6, 1.3, 2.0]$$
$$X := [746, 629, 1571, 155]$$
$$Y := [1393, 375, 259, 987]$$
$$d4 := 864.3$$

如前所述，角度和角度的标准差需要转换为弧度.

```
> for i to 3 do
>     theta[i] := evalf( 2*Pi*theta[i] / 360 );
>     if theta[i] > evalf(Pi) then
>         theta[i] := theta[i] - evalf(2*Pi) fi;
>     sigma[i] := evalf( 2*Pi*sigma[i] / 360 );
> od:
> i := 'i':
> print( theta ); print( sigma );
```

$$[2.813470755, .7871434929, -.890117918]$$
$$[.01396263402, .01047197552, .02268928028, 2.0]$$

现在我们可以构成平方和.

```
> S := sum( ( (arctan( x-X[i], y-Y[i] ) - theta[i] ) /
>                sigma[i] ) ^ 2, i = 1..3 ) +
>           ( ( ( (x-X[4])^2 + (y-Y[4])^2 ) ^ (1/2) - d4 ) /
>                sigma[4] ) ^ 2;
```

$$\begin{aligned} S := {} & 5129.384919\,(\arctan(x-746,\,y-1393)-2.813470755)^2 \\ & + 9118.906513\,(\arctan(x-629,\,y-375)-.7871434929)^2 \\ & + 1942.488964\,(\arctan(x-1571,\,y-259)+.890117918)^2 \\ & + .2500000000\,(\sqrt{(x-155)^2+(y-987)^2}-864.3)^2 \end{aligned}$$

下面我们令导数等于零并求解.

```
> sol := fsolve( { diff(S,x)=0, diff(S,y)=0 }, {x,y} );
```

$$sol := \{x = 978.3070298,\ y = 723.9837773\}$$

解被求出来了，而且它的确在我们所期望的范围内.

这一次近似是成功还是失败要确定最小二乘近似的剩余量. 在误差为正态分布的假设下，它将是四个 $N(0,1)$ 变量的平方和. 这样的平方和的期望值是 4.

```
> S0 := evalf( subs( sol, S ));
```

$$S0 := .6684712637$$

这个数值小于 4, 因此它表明或者是我们的运气很好，或者我们把误差估计太高了. 无论如何，对于近似的效果来说这是一个好消息. 它与下一节所做的特征值分析一起说明：我们得到了正确的结果.

26.4 信度 / 灵敏度分析

我们用研究解的灵敏度的程序，把误差平方和在它的最小点展开成 Taylor 级数. 令 $S(x,y) = S(\vec{p})$ 是平方和，我们将把它定义为位置向量 $\vec{p} = [x,y]^{\mathrm{T}}$ 的函数. 令 $\vec{p}_0 = [978.30\ldots, 723.98\ldots]^{\mathrm{T}}$ 是最小二乘问题的解. 则在 $\vec{p}_0$ 附近 Taylor 级数的前三项是

$$S(\vec{p}) = S(\vec{p}_0) + S'(\vec{p}_0)(\vec{p}-\vec{p}_0) + (\vec{p}-\vec{p}_0)^{\mathrm{T}}S''(\vec{p}_0)(\vec{p}-\vec{p}_0) + O(|\vec{p}-\vec{p}_0|^3).$$

因为 S 的梯度 $S'(\vec{p}) = [S'_x, S'_y]^{\mathrm{T}}$ 在最小点是零 (数值验证显示这个梯度在 $[0,0]$ 的舍入误差之内),

```
> S1 := evalf( subs( sol, linalg[grad]( S, [x,y] )));
```

$$S1 := [.36\,10^{-7}, .4\,10^{-8}]$$

略去高阶项的表达式是

$$S(\vec{p}) = S(\vec{p}_0) + (\vec{p}-\vec{p}_0)^{\mathrm{T}}S''(\vec{p}_0)(\vec{p}-\vec{p}_0).$$

在 MAPLE 中依据未知的 x 和 y 来计算 S 的近似值，首先要计算 Hessian 矩阵 (S 的二阶导数矩阵), 在最小点估计它的值，然后计算二次型.

```
> S2 := evalf( subs( sol, linalg[hessian]( S, [x,y] ) ));
> pmp0 := [ x-subs(sol,x), y-subs(sol,y) ];
> Sapprox := S0 + evalm( transpose(pmp0) &* S2 &* pmp0 );
```

$$S2 := \begin{bmatrix} .5118424680 & -.1726069278 \\ -.1726069278 & .09026159226 \end{bmatrix}$$

$$pmp0 := [x - 978.3070298, y - 723.9837773]$$

$$\begin{aligned} Sapprox := {} & .6684712637 \\ & + (.5118424680\,x - 375.7744690 - .1726069278\,y)\,(x - 978.3070298) \\ & + (-.1726069278\,x + 103.5146424 + .09026159226\,y)\,(y - 723.9837773) \end{aligned}$$

此后我们将在测量误差符合正态分布的假设下工作. 作为来自于四个 $N(0,1)$ 变量的平方和，S 将服从自由度为 4 的 χ^2 分布. 知道这个分布，就允许我们确定 $S(\vec{p})$ 的置信区间. 假设我们对 95% 的置信区间有兴趣，则有 $S(\vec{p}) < 9.4877\ldots$, 这个数值是由 χ^2 分布表中查出的,

```
> stats[ statevalf, icdf, chisquare[4] ] ( 0.95 );
```

$$9.487729037$$

不等式 $S(\vec{p}) < 9.4877\ldots$ 定义了一个椭圆，x 和 y 的真值以 95% 的概率落在椭圆内. 对于 50%, 95%, 99.9% 三个不同的置信区间我们可以画出三个椭圆，它们都是统计计算的参考值. 要说明的是置信度越大，椭圆就越大 (见图 26.2).

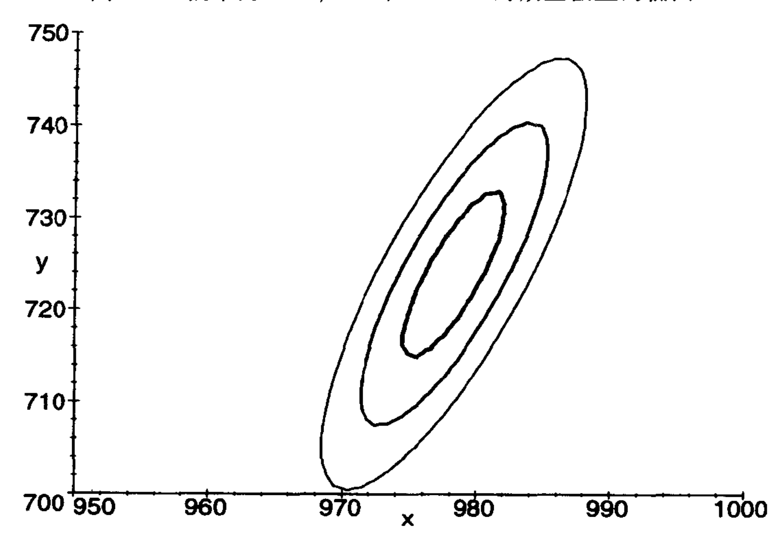
椭圆的主轴是不确定性最大的方向. 次轴是不确定性最小的方向. 这两个轴的准确的方向由 S'' 的特征值 / 特征向量分解给出.

```
>  ev := linalg[eigenvects]( S2 );
```

$$ev := [.5734960798, 1, \{[-.9417276012, .3363764637]\}],\\ [.0286079803, 1, \{[-.3363764637, -.9417276012]\}]$$

最大特征向量，0.5734... 给出了 $S(\vec{p})$ 的最陡上升方向，或者是最大信度的方向，本例中为 $[-0.941\ldots, 0.336\ldots]$. 最小的特征向量，0.0286... 给出了最小信度的方向，显然它垂直于前者. 对于这个特定的例子，我们看到 DME 与 VOR2 一起提供了大部分信息 (DME 信号台有最小的相对误差)，以及由此得到的椭圆的形状和方向.

图 26.2 概率为 50%,95% 和 99.9% 的期望位置的椭圆




```
>  ellips := { seq( stats[ statevalf, icdf, chisquare[4]](c) =
>                   Sapprox, c = [0.5, 0.95, 0.999] ) }:
>  plots[implicitplot]( ellips, x = 950..1000, y = 700..750,
>                   grid=[50,50], view=[950..1000, 700..750] );
```

计算特征值有另外一个优点. 当我们求解值导数等于零的方程时，我们可能求出最小值、最大值或者鞍点. 检验我们所得到的是最小值的正确方法是验证 Hessian 矩阵是正定的. 特征值的检查给我们提供了更多的信息. 如果所有的特征值是正的，则我们得到一个最小值，如果都是负的，则我们得到一个最大值，如果有正有负，则我们得到一个鞍点. 因为在我们的例子中，两个特征值都是正的，我们敢肯定我们求出的是最小值.

本章所描述的过程可以被推广到任意维数 (未知参数).

第二十七章　计算平面日晷

M. Oettli and H.Schilt

27.1　引言

在古教堂和一些老房子的正面常常能看到年代久远的日晷 (sundial). 一些极好的例子可以在文献 [3,4] 中找到. 这些日晷在默默地也是不引人注意地向人们诉说着那已往的岁月，当时人们正是用这些日晷测量时间. 直到 19 世纪末，甚至钟表也是靠太阳来调整的. 今天，日晷重新出现在公园里和房子的正面主要是为了装饰. 一般来说这些日晷并不十分精确. 时不时人们会看一下表来确定是否有比较大的误差. 另一方面，也有些日晷的精度令人吃惊.

有许多类型的日晷. 实际上，任何物体的投影都可以用作日晷. 本章的目的是介绍设计精确平面日晷所需要的数学知识. 这些日晷是这样的，指示物顶端的影子投射在平面上，平面刻有时间标度 (刻度盘) 以指示时间. 这个指示物称之为日晷指针 (gnomon). 这种日晷不仅能指示地方真太阳时 (local real time), 也能指示我们日常生活用的平太阳时 (mean time), 甚至还能显示其它时间.

要注意的是，为了精确，日晷必须为每个使用地点专门设计，还必须指向正确的方向. 读者用 MATLAB 的算法可以做一个自己的日晷.

本章的基础是文献 [1], 它的大致内容在后面介绍. 首先介绍必要的天文基础知识并定义后面将用到的坐标系. 日晷指针的投影问题是理解以后计算的基础. 在 27.3 节介绍了水平日规的各种时间刻度. 在 27.4 节取消了仅讨论水平日晷的限制. 本章有一个实际的例子. 在本章结束，我们介绍了有关的文章 [2].

27.2　天文基础知识

我们都知道一些天文基础知识，这是设计日晷所必需的. 地球绕太阳一年一转的椭圆轨道运动是满足 Kepler 定律的. 在轨道上，地球绕自身的轴稳定地旋转 (自转), 轴与绕太阳的轨道平面倾斜. 这个轨道平面叫黄道 (ecliptic). 白天与黑夜的变化是由于自转，而公转则带来季节的变化. 白天的突出现象是日出与日落，还有就是日中天，也就是太阳处在它的最高位置的时候 (正午).

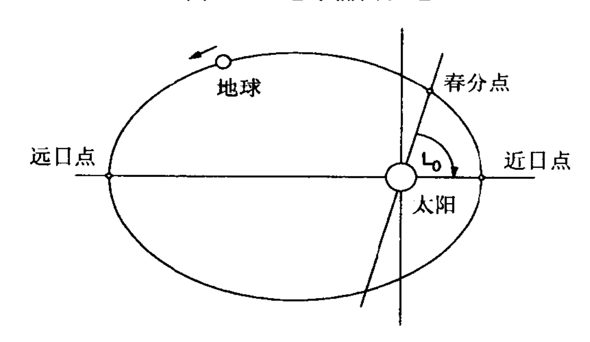
太阳位于地球椭园轨道的一个焦点上，见图 27.1. 当地球远离太阳时，它的速度逐渐变慢，直到它达到远日点 (aphelion). 在该点，地球的运动最慢，这时地球位于夏至点. 太阳的引力开始把地球拉向太阳并增加它的速度. 太阳不断地给地球加速，直到地球达到轨道上相反的顶点，近日点 (perihelion). 在春分点 (spring equinox), 白天与黑夜的长度相等.

我们需要的一个天文常数是地球椭园轨道离心率的数值. 这个值是 $e = 0.01672$. 我们还需要黄道与赤道平面的夹角 $\epsilon = 23.44°$, 最后一个值是近日点的黄经 (ecliptic longitude) $L_0 = -77.11°$, 它是从春分点逆时针开始计算. 实际上这些常数并不十分精确. 例如，近日点大约 21,000 年绕太阳旋转一周. 在以下的计算中将忽略这些.

27.2.1　坐标系

天文学家使用着不同的坐标系. 它们都有一组正交向量基 $(\mathbf{x}_a, \mathbf{y}_a, \mathbf{z}_a)$. 如果 $\mathbf{s}$ 是一个空间的向

图 27.1 地球椭圆轨道



量， $\mathbf{s}_a$ 代表它在基 $(\mathbf{x}_a, \mathbf{y}_a, \mathbf{z}_a)$ 下的向量表示.

地平坐标系 (horizontal coordinate system) 如图 27.2 所示. 观测者位于地平面的原点. 观测者垂直上方的点叫天顶 (zenith), 在球体相反的对应点叫天底 (nadir). 仰角 h 是恒星与地平面的夹角. 负值表示位于地平面以下. 方位角 a 是南极点 S 与天顶的大园和经过恒星与天顶的垂直园之间的夹角.

图 27.2 地平系

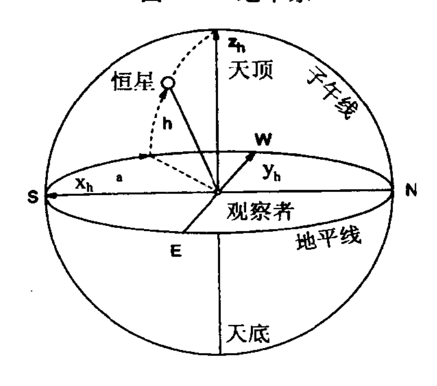
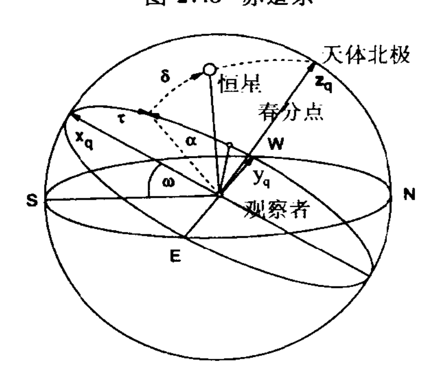


图 27.3 赤道系

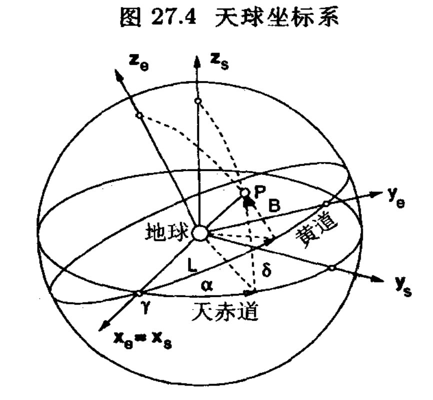


地平坐标系的缺点是它要以天顶为参考点，而这点在宇宙中并不固定. 对天文观测者而言，更常用的是局部赤道坐标系 (local equatorial system), 它是用北极和当地赤道为参考，见图 27.3. 恒星的时角 τ 是顺着每日运动的方向沿赤道测量子午线与恒星的夹角. （子午线是通过南极和北极的大园). 恒星的赤纬 (declination) δ 是它到赤道的角距. 恒星位于赤道以北为正，反之为负. 常用赤经 (right ascension) α 取代时角 τ, 它是从春分点开始沿赤道向正方向测量.

还有两个天体坐标系常用来描述地球绕太阳的年运动，两个坐标系都是地心系，它忽略了地球的自转而专注于太阳相对于地球的位置. 与上面的坐标系不同，它们与地球的特定位置无关.

春分与秋分点是黄道与天体赤道的交点. 春分点在图 27.4 上用 γ 表示. 一方面，我们有了天赤道坐标系 (system of the celestial equator), 它用两个角 δ 和 α 表示太阳的位置. 从天赤道到太阳的角距是赤纬 δ. 春分点与地球自转轴组成的平面和太阳与地球自转轴组成的平面之间的夹角是赤经 α. 而在黄道坐标系中，太阳的位置是由它在黄道上从春分点开始计算的角度来表示. 这个角叫黄经 L. 一个任意点 P 还可以进一步用黄纬 (ecliptic latitude)B 来表示 (往北极的方向为正).

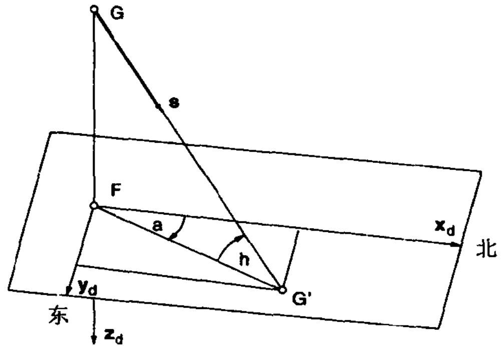
图 27.4 天球坐标系



27.2.2 日晷指针的投影

考虑日晷指针的顶端投影在水平面上，见图 27.5. 在以下的坐标系中，向量 $\mathbf{s}$ 表示日晷指针顶端影子的投影方向. 从日晷指针的顶端 G 向平面作垂线，垂足 F 为坐标系的原点. $\mathbf{x}_d$ 轴向北，$\mathbf{y}_d$ 轴向东，$\mathbf{z}_d$ 轴在垂线方向. 在这个坐标系中，向量 $\mathbf{s}$ 表示为

图 27.5 表盘坐标系



$$\mathbf{s}_d = \begin{pmatrix} \cos(h)\cos(a) \\ \cos(h)\sin(a) \\ \sin(h) \end{pmatrix},$$

这里 a 是方位角，h 是纬度. 接下来，日晷指针顶端的影子 G' 在平面上的坐标是

$$g_1' = \frac{s_1}{s_3}g,\ \ g_2' = \frac{s_2}{s_3}g,\ \ g_3' = 0,$$

其中，g 是日晷指针的高度，即距离 GF.

使用 27.2.1 节的地平坐标系，向量 s 可以表示成

$$\mathbf{s}_h = \begin{pmatrix} \cos(h)\cos(a) \\ \cos(h)\sin(a) \\ \sin(h) \end{pmatrix} = -\mathbf{s}_d.$$

现在要讨论的问题是，已知太阳在赤道坐标系中的位置，如何决定日晷指针顶端的影子在日规盘面上的位置. 太阳的位置 $\mathbf{v}$ 依赖赤纬 δ 和时角 t, 写成

$$\mathbf{v}_q = \begin{pmatrix} \cos(\delta)\cos(t) \\ \cos(\delta)\sin(t) \\ \sin(\delta) \end{pmatrix}.$$

把赤道坐标变换成地平坐标要进行一次平面转动，见图 27.3. 我们得到

$$\mathbf{v}_h = \begin{pmatrix} \cos(w) & 0 & -\sin(w) \\ 0 & 1 & 0 \\ \sin(w) & 0 & \cos(w) \end{pmatrix} \mathbf{v}_q,$$

这里 $w = 90° - \phi$,ϕ 是观测者所处的纬度. 因而，给定太阳的时角 t 和赤纬 δ, 当地的纬度 ϕ 和日晷指针的长度 g, 就可以用 MATLAB 给出的算法 27.1 来计算影子.

算法 27.1 函数 project

```
function project(t,d,phi,g,cmd)

X = [cos(d).*cos(t);           % rays in equatorial sys.
     cos(d).*sin(t);
     sin(d).*ones(size(t))];
w = pi/2 - phi;
R = [cos(w)  0  -sin(w);
      0      1    0;
     sin(w)  0   cos(w)];
X = R*X;                       % equatorial -> horizontal
ix = (X(3,:) > 0);             % only rays from above
if any(ix),
  X = g*X(1:2,ix)./(ones(2,1)*X(3,ix));
                               % shadow points
  plot(X(2,:),X(1,:),cmd);
end
```

顶点的影子 G' 的计算表示了上半球在平面上的逐点投影. 这个图叫日晷指针投影. 大园成了一条直线，而另外的园则成了园锥型的曲线. 尤其特别的是，赤道也画成了一条直线. 赤纬不为零时，太阳每日的拱线均成了圆锥曲线，称之为赤纬线 (太阳的赤纬沿着给定的每日拱线实际是一个常数). 园锥曲线的类型取决于纬度 ϕ 和赤纬 δ.

27.3 时间标度

前一节的基础知识使我们在知道赤纬 δ 和时角 t 后，能确定日晷指针顶端在水平面的影子. 现在来讲各种有趣的时间标度. 从午夜开始，时间分为 24 小时. 注意有一些老的日晷可能从日中天的时候即正午开始计算.

27.3.1 地方真太阳时

直接由太阳的运动来决定的时间叫地方真太阳时或叫地方视时. 当太阳在白天的最高点时，这就是当地时间的正午. 如果两个地方不在同一个子午线上，它们的地方时是不同的. 因而这个时间定位在一个特定的子午线上. 此外，一年之中地方时各个小时的长度是不同的. 由于地球绕太阳的椭园轨道和地球自转轴对黄道面的倾斜，我们看到的太阳在天空中的运动在一年之中也是不相同的. 日晷所显示地方真太阳时与钟表的时间相比可能快慢 16 分钟. 见 27.3.2 节.

对地方真太阳时来说，小时线是一个特殊的例子. 时角 t_r 与赤纬无关，表示为

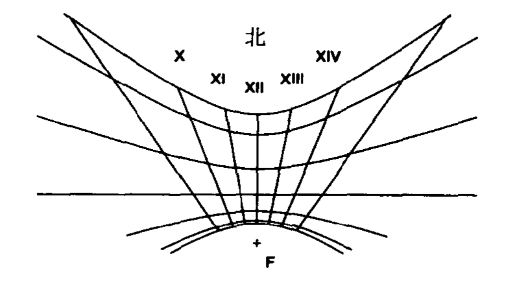
$$t_r = 15°(k - 12).$$

小时线是大园的日晷指针投影，因此在图 27.6 所示的表盘中，它被画成一条直线. 此外，图 27.6 的表盘画了七条赤纬线，它的刻度表示黄道十二宫 (zodiac) 的一个新宫的开始. 按照一个古老的巴比仑传统，黄道被划分为十二个相等的区间，每个区间为 30°. 表 27.1 给出了十二宫开始的赤纬 δ 和黄经 L. 这些值很容易从黄道坐标系和天体赤道坐标系的关系导出，见 27.2.1 节.

表 27.1 黄道十二宫

L	sign	δ	L	sign	δ
0°	Aries	0.0°	180°	Libra	0.0°
30°	Taurus	11.47°	210°	Scorpio	−11.47°
60°	Gemini	20.15°	240°	Sagittarius	−20.15°
90°	Cancer	23.44°	270°	Capricorn	−23.44°
120°	Leo	20.15°	300°	Aquarius	−20.15°
150°	Virgo	11.47°	330°	Pisces	−11.47°

图 27.6 太阳时和 +47° 纬度的地平日晷



下面摘录的程序使用 MATLAB 程序 `project`, 在给定纬度 ϕ 后，画出地方真太阳时的刻度盘.

```
>> g = 1; eps = 23.44; p = pi/180;
>> cls; axis([-5 5 -2 5]); hold on;
```

```
>> phi = 47*p;                  % latitude
>> k = [7:17];
>> t = 15*p*(k-12);
>> d = p*[23.44 20.15 11.47 0 -11.47 -20.15 -23.44];
>> for i=1:length(d),           % declination lines
>>    project(t,d(i),phi,g,'-');
>> end
>> d = eps*p*[-1 1];
>> for k=8:16,                  % hour lines
>>    t = 15*p*(k-12);
>>    project(t,d,phi,g,'-');
>> end
```

27.3.2 平太阳时

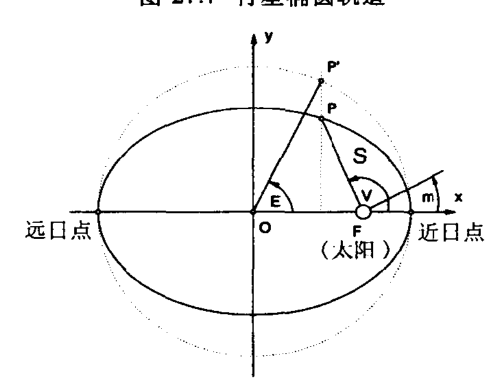
日常生活中使用的是平太阳时. 不论什么季节, 所有的小时都有相同的长度. 平太阳时使时钟的划分成比例. 地方平太阳时定义为假想的太阳以不变的速度在绕天体赤道而不是绕黄道前进, 这个速度是真实太阳在黄道上运动的平均速度. 假想的太阳与真实太阳在同一时刻离开春分点 γ, 在一个回归年后又同时回到这点.

为了在日晷上引入平太阳时, 必须对地方真太阳时作个修正. 这个修正定义为真太阳时与平太阳时之差, 叫做时间方程. 在设计平太阳时的时间线之前, 在下两节先引入时间方程.

Kepler 方程

行星运动的 Kepler 定律是引入平太阳时的基础. 按照第一个定律, 地球轨道是以太阳为一个焦点的椭园, 见图 27.7.

图 27.7 行星椭圆轨道



轨道上 P 点的坐标是

$$\mathbf{x} = (a\cos(E), b\sin(E)),$$

a 和 b 分别是椭园的长半轴和短半轴. 角 E 叫偏近点角 (eccentric anomaly). F 焦点到零点 O 的距离是 ae, e 是偏心率. 参数 e 决定了椭园的形状. 现在地球轨道的偏心率 $e = 0.01672$. 而角 v 叫

真近点角 (true anomaly). 它是以 F 为原点的坐标系的极角. 给定 F 点与 P 点的坐标, 就可以计算向量 FP 的长度 r. 它是

$$r = a(1 - e\cos(E)), \qquad r\cos(v) = a(\cos(E) - e).$$

第一个方程利用了椭圆是虚线圆的仿射映射这个事实而得到. 后一个方程是在 x 轴上距离的简单比较. 对这两个方程经过一些冗长的计算之后, 得到 v 和 E 的关系:

$$\tan(E/2) = \tan(v/2)\tan(\arccos(e)/2). \tag{27.1}$$

行星的运动使得 v 和 E 的值不规则变化. 而按照 Kepler 第二定律, 矢径 FP 在等时间间隔内扫过相等的面积. 因而, 在图 27.7 中阴影面积 S 有规律地增长. 借助于仿射映射, 可得到

$$S = \frac{ab}{2}(E - e\sin(E)),$$

E 用弧度表示. 括号中的量叫平均近点角 (mean anomaly), 它正比于时间. 我们把它简写为 m. 由此得到 Kepler 方程.

$$m = E - e\sin(E), \tag{27.2}$$

它把偏近点角 E 和平均近点角 m 联系起来. 平均近点角是一个假想的行星在从太阳看到的近日点的角距. 这个假想的行星绕太阳运转的速度是常数, 运转的时间与真实的行星相同.

时间方程

由于地球在绕日轨道上的速度不断变化, 这是 Kepler 第二定律所描述的情况, 以及地球自转轴相对于黄道面的倾斜, 太阳在天空中的运动在一年之中是不一致的. 故要对地方真太阳时进行修正, 以表示平太阳时. 这个修正叫做时间方程.

地球速度变化对时间方程的影响 z_k 在真近点角和平均近点角之间是不同的, 即

$$z_k = m - v.$$

用真实太阳的黄经 L 计算这个量最为理想. 记住 L 是从春分点开始计算的, 见图 27.2.1. 另一方面, 实际近点角 v 是从近日点开始计算, 现在的经度是 $L_0 = -77.11°$, 见图 27.1. 因此实际的近点角 v 是

$$v = L - L_0.$$

然后, 用方程 27.1 和 27.2 计算偏近点角 E 和平均近点角 m.

对时间方程的第二个影响是由于地球自转轴对黄道面的倾斜. 这个倾斜角为 $\epsilon = 23.44°$, 并且总是指向同一个方向. 如果地球自转轴垂直于黄道面, 则不必修正. 因此是假想的平太阳时定义了平太阳时, 它沿天体赤道运行. 结果是, 对时间方程的影响 z_t 在经度 L 和真实太阳的赤经之间是不同的. 即:

$$z_t = L - \alpha.$$

赤经 α 是从 L 利用真实太阳在两个天体坐标系中的关系式

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\epsilon) & -\sin(\epsilon) \\ 0 & \sin(\epsilon) & \cos(\epsilon) \end{pmatrix} \begin{pmatrix} \cos(L) \\ \sin(L) \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\delta)\cos(\alpha) \\ \cos(\delta)\sin(\alpha) \\ \sin(\delta) \end{pmatrix}$$

图 27.8 时间方程作为黄道十二宫的函数

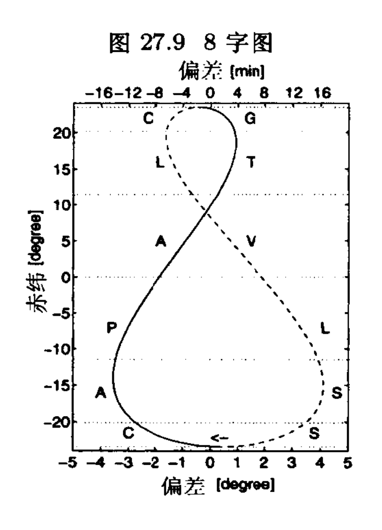
得到的.

时间方程 z_g 是 z_k 与 z_t 之和. 由于 α 在 $L = 180°$ 发生跳跃, E 在远日点 ($v = 180°$) 发生跳跃, 必须令 $z_g = \arctan(\tan(z_g))$, 以保持 z_g 的连续性. 因此时间方程的观测值为

$$z_g = \arctan(\tan(z_k + z_t)).$$

图 27.8 画出了时间方程 z_g 以及它的两个部分 z_k 和 z_t 相对于黄道十二宫的曲线 (由于闰年的调整, 在不同年份 z_g 每天的变化很小). 当 z_k 在地球轨道的近日点和远日点为零时, z_t 处在季节的开始, 即春分点和秋分点 (6 月 21 日和 12 月 21 日) 也为零. 图 27.9 表示太阳赤纬对时间方程的依赖关系. 这个图形象数字 8, 称为 8 字图. 如果在一年之内每天都在同一个时间里画太阳的位置, 得到的结果就是一个 8 字图. 实线是时间从 12 月 21 日到 6 月 21 日, 间断线则是一年的另一半时间. 圆点的赤纬线刻度表示黄道十二宫的一个新宫的开始.

图 27.9 8 字图



当地平太阳时

当地平太阳时是由前一节引入的假想平均太阳的时角所定义的时间. 当地平太阳时 T_m 等于当地真太阳时 T_r 减去时间方程,

$$T_m = T_r - z_g.$$

当地平太阳时的第 k 个小时的时角 t_m 是

$$t_m = 15^\circ(k-12) + z_g.$$

用 8 字图标上平太阳时的每个小时的刻度, 可以直接从刻度盘上读出平太阳时而不必做任何修正. 下面摘录的程序首先计算了一年的太阳的赤纬和时间方程. 用这些信息很容易画出第 k 个小时的 8 字图.

```
>> L0 = -77.11;  e = 0.01672;  eps = 23.44;  p  = pi/180;
>> L  = 3*p*[-30:90];               % Sun's True Longitude
>> v  = L - L0*p;                   % True Anomaly
>> c  = sqrt((1-e)/(1+e));          % c=tan(arccos(e)/2)
>> E  = 2*atan(c*tan(v/2));         % Eccentric Anomaly
>> zk = E-e*sin(E)-v;
>> x  = [cos(L);                    % Sun's Coordinates in System
>>       sin(L)*cos(eps*p);         % of Celestial Equator
>>       sin(L)*sin(eps*p)];
>> r  = sqrt(x(1,:).^2+x(2,:).^2);
>> al = atan2(x(2,:),x(1,:));       % Right Ascension
>> d  = atan2(x(3,:),r);            % Declination
>> zt = L-al;
>> zg = atan(tan(zk+zt));           % Equation of Time
>> for k = 9:15,                    % Individual Hour Lines
>>   t  = 15*p*(k-12)+zg;
>>   project(t,d,phi,g,'-y');
>> end;
```

时区

迄今为止讨论的地方平太阳时是对一个子午线在天文学上的修正. 一个在西边远方的日晷显示的时间要落后于当地的日晷. 而一个时间统一的系统在日常生活中才是实用的. 所以, 在 19 世纪末便引入了时区或标准时. 整个地球划分 24 个时区, 从本初子午线开始, 每个区的经度宽度约为 15°. 本初子午线的时间叫格林威治时间 (GMT, Greenwich Mean Time), 从它向东, 每个时区增加一个小时, 向西则每经过一个时区减少一个小时.

时区的时间与地方平太阳时间的不同之处为当地子午线 λ 与时区子午线 λ_0 在经度上的差别. 因此第 k 个时区的时角为

$$t_z = 15^\circ(k-12) + z_g + (\lambda_0 - \lambda).$$

27.3.3 巴比仑时与意大利时

我们认为是巴比仑人将一天分为 24 小时，他们习惯从太阳升起的时候开始计时．与此相反，中世纪的意大利人是把日落当作一天的开始．这两个计时系统都沿用了很长时间，即使在现在也能发现无数的古老欧洲日晷，它们有的用巴比仑时有的用意大利时．尽管它们已经不再使用，但可以用这些时间刻度来研究日规．令

$$\mathbf{u}_q = \begin{pmatrix} \cos(\delta)\cos(\tau) \\ \cos(\delta)\sin(\tau) \\ \sin(\delta) \end{pmatrix}$$

是日落时的太阳位置．向量 $\mathbf{u}$ 在地平面内．因而

$$\mathbf{u}_h = \begin{pmatrix} \cos(w) & 0 & -\sin(w) \\ 0 & 1 & 0 \\ \sin(w) & 0 & \cos(w) \end{pmatrix} \mathbf{u}_q,$$

其中 $w = 90° - \phi$, 而纬度为 ϕ 的 z 分量必须为零．由此得到日落时的时角 τ 的方程

$$\cos(\tau) = -\tan(\delta)\tan(\phi).$$

负解 $t = -\tau$ 是日出时的时角．注意 τ 也是白天长度的一半．第 b 个巴比仑小时与第 i 个意大利小时的时角分别是

$$t_b = 15°b - \tau \quad 和 \quad t_i = 15°i + \tau$$

时角 t 不仅依赖于 b(或者 i), 也依赖于太阳的赤纬．回想一下日晷指针投影法：巴比仑时和意大利时的小时线都是大圆的投影所以都是直线．因此第 b 个巴比仑与第 i 个意大利的小时线的交点位于 $t_r = (b+i)/2$ 的地方真太阳时的小时线上．

对于纬度 $\phi > |90° - \epsilon|$, 巴比仑时和意大利时只对赤纬定义，对此既没有极点的白日也没有极点的黑夜．这意味着它们只定义在 $\delta \in [-\delta_0, \delta_0]$ 内，其中

$$\delta_0 = \begin{cases} \epsilon & : \quad |\phi| + \epsilon \le 90° \\ 90° - \phi & : \quad 其它情况. \end{cases}$$

下面摘录的程序画出了巴比仑小时线．每个小时线都计算了三个点，它们应该在一条直线上．

```
>> % Babylonic hours
>> if (abs(phi)+eps*p > pi/2),   % declination restricted above
>>   d0 = pi/2-phi;               %  polar circle
>> else
>>   d0 = eps*p;
>> end
>> s0 = acos(-tan(d0)*tan(phi));
>> k3 = fix(s0/7.5/p);           % number of hours for longest day
>> d = d0*[-1:1];
>> s = acos(-tan(d)*tan(phi));
```

```
>> for k=0:k3,
>>   t = 15*k*p-s;
>>   project(t,d,phi,g,'-.r');
>> end
```

图 27.11 给出了有巴比仑小时线与意大利小时线的日晷.

27.4 任意平面上的日晷

迄今为止，我们讨论的日晷是在地平面上. 把它推广到任意方位的平面上并不困难. 例如，任意平面的方向由平面垂线 $\mathbf{n}$ 的方向角 a 和仰角 h 来定义，见图 27.10.

图 27.10 地平面和任意平面

有两种不同方法来计算任意平面的日晷. 第一种, 用两个旋转变换把水平日晷的坐标系 $(\mathbf{x}_d, \mathbf{y}_d, \mathbf{z}_d)$ 变换成任意平面日晷的坐标系 $(\mathbf{x}'_d, \mathbf{y}'_d, \mathbf{z}'_d)$. 第二种，我们可以在地球上确定一个位置，对这一点而言，它的方位角平行于所需平面的垂线. 然后我们可以计算这个新位置的水平日晷. 这个方法需要球面三角的一些知识并需修正现有程序 [1]. 首先让我们用第一种方法.

利用两次旋转把在水平日晷的当地坐标系中的一条太阳光线 $\mathbf{v}_d$ 变成任意平面的坐标系中的一条太阳光线 $\mathbf{v}'_d$. 具体来说，向量 $\mathbf{v}'_d$ 由

$$\mathbf{v}'_d = \begin{pmatrix} \cos(b) & 0 & -\sin(b) \\ 0 & 1 & 0 \\ \sin(b) & 0 & \cos(b) \end{pmatrix} \begin{pmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{v}_d.$$

决定，这里 a 是南极点顺时针测量的方位角，而 $b = 90° - h$, h 为仰角. 相应的对程序 `project` 的修正是十分方便的，我们把它留给读者.

27.5 计算实例

本章给出的代码很容易加到 MATLAB 中去，它会调用画日晷的程序 `project`. 将输出适当放大，就可以给一位熟练的画匠当作制作精确日晷的草图. 程序要求输入的是精确的地理位置，日晷平面的方向和它的大小，大小决定了日晷指针的长短. 如果日晷还要显示时区的时间，则还需要时区子午线的经度 λ_0. 地理参数必须相当仔细才能保证精度. 最后的困难是校正日晷指针的位

置．读取时间使用的只是它的顶端，它必须置于日晷坐标系零点上方的预定距离．日晷指针本身可以有任意的方向，但如果需要把地方真太阳时读出来，它必须平行地球的自转轴．要想更详细地了解如何设计一个日晷请参考文献 [1].

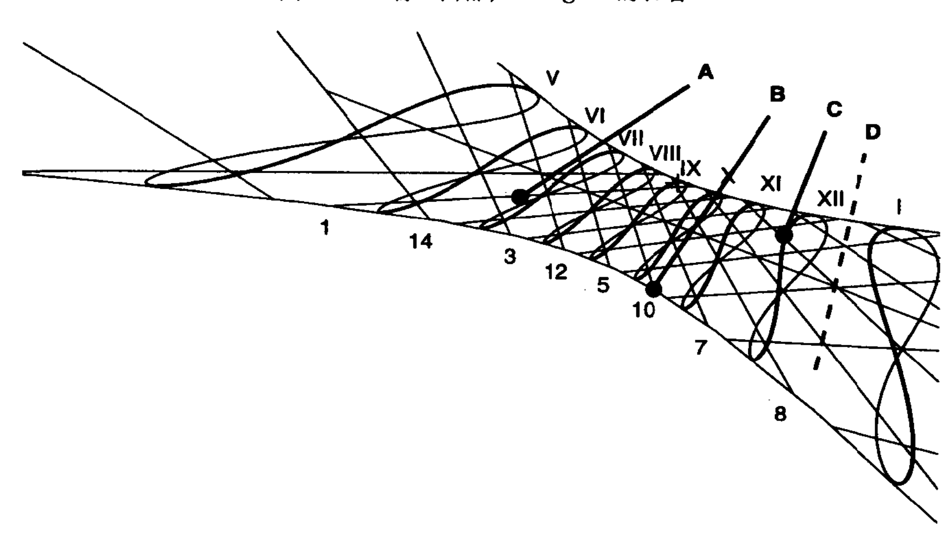
图 27.11 所示的是在瑞士西部的 Vingelz 的一个几乎垂直墙上的日晷．日晷的参数如表 27.2 所示．

表 27.2 地理位置和方位

经度	−7°13′01″	方位角	−48.95°
纬度	47°07′48″	仰角	5.71°

日晷指示了欧洲中心时 (CET, Central European Time), 还有巴比仑时和意大利时．甚至也能读出地方真太阳时，只要日晷指针平行于地球的自转轴．在 8 字图后面的直线网格给出了巴比仑时和意大利时．每隔一条线都标上了阿拉伯数字．直线 1,3,5,7 表示自从太阳升起以后已经过了多少个小时 (巴比仑时), 而直线 8,10,12,14 表示到日落还有多少个小时 (意大利时)．为了读出时间，请看图中的几个大黑圆点．它代表日晷指针顶端的影子．

图 27.11 瑞士西部， Vingelz 的日晷



A. 从太阳升起已经过了多少个小时？ 请看那条从左下到右上的稍微倾斜的直线．例 A 表示，影子所指示的时间或者是 4 月 20 日 6 点 40 分 (CET) 或者是 8 月 23 日 6 点 45(CET). 它是太阳升起后的一小时．

B. 到日落还有多少小时？ 请看那条从左上到右下的很陡的直线．例 B 表示影子所指示的时间是 6 月 21 日 10 时 20 分 (CET) 上：到日落还剩下 10 个小时．

C. 现在是什么时间？ 请看那些带有罗马数字的 8 字图．当影子投射在 8 字线上，它正好是一个整的小时 (CET). 8 字图比较粗的部分对应前半年 (12 月 21 日 –6 月 21 日), 比较细的部分对应后半年 (6 月 21 日 – 12 月 21 日). 例 C 表示影子所指示的时间是 2 月 19 日 12 时 (CET).

D. 地方真太阳时． 巴比仑与意大利时线的交点可用来确定地方真太阳时．为此，只要看此时日晷指针的影子．例 D 表示，影子所指示的时间为地方真太阳时间正午 12 时．对 Vingelz 来说，

太阳处在白天的最高点.

参考文献

[1] H. SCHILT, *Ebene Sonnenuhren: verstehen und planen, berechnen und bauen*, 9. Auflage, Felix Solis Tempus, Basel, 1997.

[2] CH. BLATTER, *Von den Keplerschen Gesetzen zu einer minutengenauen Sonnenuhr*, Elemente der Mathematik (49), 155-165, 1994.

[3] R. ROHR, *Sundials: History, Theory, and Practice*, University of Toronto Press, Toronto, 1970.

[4] A. ZENKERT, *Faszination Sonnenuhr*, Verlag Harri Deutsch, 2. Auflage, Thun, 1995.